CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

A Component Framework for Personalized Multimedia Applications

Dissertation zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften

von

Dipl.-Inform. Ansgar Scherp

Gutachter:

Juniorprof. Dr. Susanne Boll
Prof. Dr. Ralf H. Reussner

Tag der Disputation: 25. August 2006

Acknowledgment

I am deeply indebted to many people who have provided me help and support for my work and research in the past years. Thus, I like to take this as opportunity to thank them.

First of all, I would like to thank Prof. Dr. Susanne Boll for being my supervisor of the thesis. You always supported me throughout the last three years. I very much appreciate the plenty and sometimes long discussions about multimedia issues we had. Thank you for giving me the opportunity to getting to know interesting research topics and issues. It is a pleasure to working with you and I very much enjoyed my time being your PhD student.

Second, I like to thank Prof. Dr. Ralf Reussner for the support you gave me and for being my secondary supervisor. I very much appreciate the time we spent with discussing the software engineering issues of my work. You provided me many very valuable comments and input in regard of component technology and software frameworks. I very much enjoyed the time in the working group "software architectures" of the German Association for Computing Science, writing a German handbook to software architecture.

I would like to thank Prof. Dr. Hans-Jürgen Appelrath for his long-term support during my time at the University of Oldenburg and the Oldenburg R&D Institute for Computing Tools and Systems (OFFIS). I very much thank you for giving me the chance to participate in a group study exchange program to Japan in 2004. It has been a unique opportunity to gain insight into Japanese culture, which I will never forget.

I thank Jochen Meyer, the director of the R&D division Multimedia and Internet Information Services at OFFIS, for the support you gave me during my PhD research. You always have been flexible and supporting me in regard of my "working requirements", be it travels to conferences, my research exchange to the Technical University of Vienna, or time scheduling issues. I very much appreciate that.

Special thanks go to Holger Cremer who has been my scientific research assistant for the last years and accompanied developing the MM4U framework almost from its beginnings. Thank you for the good and always reliable work you provided such as developing the binary Flash presentation format generator, improving the MM4U framework in many aspects, advancing the context-driven smart authoring tool *x*SMART, and developing new applications such as the photo album authoring wizard. Moreover, I would also like to thank my former colleague Dr. Jens Krösche and former scientific research assistant Daniel Thobe for developing the thick-client variant of Sightseeing4U, a mobile tourist guide application for our hometown Oldenburg in Germany. I thank Dr. Ross King and Niko Popitsch of the Digital Memory Engineering (DME) group at the Research Studios in Austria for providing me support with integrating the DME's multimedia database METIS into my work. I also thank you for integrating and employing my Transformation4U service into your multimedia database METIS allowing the database for a multichannel-delivery of multimedia presentations. It was a pleasure to collaborating with you.

Further, I thank Matthias Nitsche for developing the XML-based user profile information server in your Diploma thesis. I like to thank Joachim Gelhaus and Wilko Heuten for developing the Manual4U application providing interactive instruction manuals on mobile devices. Thanks go to Christine Seeliger for developing

the Sports4U application within your Bachelor thesis. It added a personalized multimedia sports news ticker to the MM4U framework's application portfolio. Thank you Malte Mathiszig for being my scientific research assistant and further improving the *x*SMART authoring tool. I thank Marko Hoyer for developing the initial version of the *x*SMART authoring tool within the context of the LEBONED project of integrating e-learning content into digital libraries. I hope you are pleased to observe the improvements made to this tool by Holger and Malte. Further thanks go to many other colleagues at the OFFIS and University of Oldenburg for the discussions and feedback you provided me.

Finally, I would like to thank my parents for supporting me and giving me the opportunity to study. I very much appreciate your esteem for education and family. I also very much thank my brothers and sisters and my friends for your support during writing this thesis. Your encouragement and time we spend together very much helped me with finishing this work. In particular, I like to thank my sister Gudrun for proofreading the thesis.

Dankeschön!

Ansgar Scherp, Oldenburg, 2007

Abstract

Multimedia content today can be considered as the composition of media elements in time and space into a coherent, interactive multimedia presentation. Personalization of such multimedia content means that it needs to reflect user's profile information and context information. To create such personalized multimedia content, a manual authoring of many different documents for all the different users' needs, however, is not feasible not to mention economical. Rather a *dynamic* authoring process of selecting and assembling personalized multimedia content depending on the user's profile information and context information seems reasonable. A practical support for such a dynamic authoring of personalized multimedia presentations is neither provided by industrial solutions nor research projects today. To provide a general and at the same time practical support for the dynamic authoring process, a software engineering approach is pursued with the MM4U (short for "MultiMedia For You") framework.

The aim of the MM4U component framework is to provide application developers with an extensive and domain independent support for the authoring of personalized multimedia content. The framework provides generic functionality for typical tasks of the general process chain for creating personalized multimedia content. This includes selecting media elements such as audio, video, image, and text according to the user's individual profile and context information. These media elements are then assembled and composed in time and space using an abstract multimedia content representation model. This model captures the different aspects of the multimedia presentation such as the temporal course, the spatial layout, and the interaction possibilities without instantiating these in a concrete syntax and format. Only in the subsequent transformation phase—called the *last mile*—the multimedia content is transformed into a concrete multimedia presentation format, like SMIL, SVG, and Flash, and delivered to the targeted (mobile) end device.

To reach the goal of a software framework providing generic support for developing personalized multimedia applications raises the question of a proper software engineering support to develop such a framework. A software framework, like the MM4U framework, typically constitutes a semi-finished software architecture for a specific domain. Since the introduction of object-oriented frameworks in the late eighties, the development of software frameworks is still costly and difficult to handle. To reduce development risk, process models and development methodologies have been developed. With the emergence of component technology also so-called component frameworks appeared. Component frameworks aim to overcome some drawbacks of object-oriented frameworks, such as problems that occur when combining object-oriented frameworks and changes at the bases classes of these frameworks become necessary. In contrast to object-oriented frameworks, a proper process model for component frameworks is still missing. To improve the development process of component frameworks, the dissertation contributes the ProMoCF approach (short for "Process Model for Component Frameworks"), a lightweight process model and development method for component frameworks. This process model has been developed in mutual benefit with the development of the MM4U component framework. While the ProMoCF approach has been evaluated by applying it for the development of the MM4U component framework, the component framework itself has given important feedback for improving and maturing the ProMoCF approach.

This mutual impact between the ProMoCF approach and the MM4U component framework has resulted in constant improvement of the ProMoCF approach and the MM4U framework over several iterations. For evaluating the ProMoCF approach it has not only been applied for the development of the MM4U component framework. However, the outcome of applying the ProMoCF approach, i. e., the MM4U component framework itself, has been applied and evaluated for the development of several applications in the domain of multimedia personalization.

The MM4U component framework provides generic support for the dynamic authoring of personalized multimedia content. It does not reinvent multimedia content adaptation but is targeted at incorporating and embedding existing research approaches and solutions in the field and allows for being extended by domain and application-specific solutions. With such a framework at hand, application developers can efficiently realize on-the-fly multi-channel generation of personalized multimedia content that meets the end user's requirements and overcomes the *last mile* in multimedia presentation delivery. The MM4U framework has been employed for the development of several demonstrator applications from different domains. These demonstrator applications such as a personalized mobile multimedia city guide, a personalized multimedia sports news ticker, and a personalized picture albums generator show the applicability of the framework.

Zusammenfassung

Die Erstellung von personalisierten Multimedia-Inhalten wird als Komposition von Medienelementen in Raum und Zeit in eine kohärente, interaktive Multimedia-Präsentation verstanden. Personalisierung von solchen Multimedia-Inhalten bedeutet, dass sie den Profilinformationen über den Benutzer und den Informationen über dessen Kontext genügen. Eine manuelle Erstellung von personalisierten Multimedia-Inhalten, also die Erstellung von vielen verschiedenen Multimedia-Dokumenten für die unterschiedlichen Bedürfnisse der Benutzer, ist nicht durchführbar, ganz zu schweigen von dessen Wirtschaftlichkeit. Vielmehr wird ein Prozess zur *dynamischen* Erstellung von personalisierten Multimedia-Inhalten benötigt, bei dem die Medienelemente auf Basis der Profilinformationen über den Benutzer und den Informationen über dessen Kontext ausgewählt und zusammengefügt werden. Eine praktikable Unterstützung für eine dynamische Erstellung von personalisierten Multimedia-Präsentationen bieten bisher weder industrielle Lösungen noch Forschungsansätze. Um eine generische und zugleich praktikable Unterstützung für den dynamischen Erstellungsprozess bereitzustellen, wird ein Software-technischer Ansatz mit dem MM4U-Framework (Abkürzung für "MultiMedia For You") verfolgt.

Das Ziel des MM4U-Frameworks ist es den Anwendungsentwicklern eine umfangreiche und anwendungsunabhängige Unterstützung zur Erstellung von personalisierten Multimedia-Inhalten anzubieten. Das Framework stellt dazu generische Funktionalitäten für die typischen Aufgaben im allgemeinen Prozess zur Erstellung von personalisierten Multimedia-Inhalten zur Verfügung. Dies beinhaltet die Auswahl der Medienelemente, wie zum Beispiel Audio, Video, Bilder und Texte, hinsichtlich der Informationen über den individuellen Benutzer und den Informationen über dessen Kontext. Die Medienelemente werden dann in Raum und Zeit arrangiert und zusammengefügt. Dazu wird ein anwendungsunabhängiges Multimedia-Repräsentationsmodell verwendet. Dieses Repräsentationsmodell umfasst die verschiedenen Aspekte der Multimedia-Präsentation, wie zum Beispiel den zeitlichen Verlauf und die räumliche Anordnung der Medien sowie die Interaktionsmöglichkeiten des Benutzers mit der Präsentation, ohne diese in konkreter Syntax und Präsentationsformat zu instanziieren. Erst in der nachfolgenden Transformationsphase – die so genannte *letzte Meile* – werden die Multimedia-Inhalte in die konkreten Multimedia-Präsentationsformate wie beispielsweise SMIL, SVG und Flash transformiert und zur Darstellung auf das jeweilige (mobile) Endgerät übertragen.

Um das Ziel eines Software-Frameworks zur generischen Unterstützung der Entwicklung von personalisierten Multimedia-Anwendungen zu erreichen, stellt sich die Frage nach einer geeigneten Software-technischen Unterstützung zur Entwicklung eines solchen Frameworks. Ein Software-Framework, wie das MM4U-Framework, stellt typischerweise eine halbfertige Software-Architektur für eine bestimmte Domäne dar. Seit der Einführung von objektorientierten Frameworks Ende der 80er Jahre, ist heute die Entwicklung von Software-Frameworks immer noch aufwendig und schwierig zu bewältigen. Um die Entwicklungsrisiken zu reduzieren, sind geeignete Vorgehensmodelle und Entwicklungsmethoden erstellt worden. Mit der Entwicklung der Komponenten-Technologie sind auch so genannte Komponenten-Frameworks entstanden. Ziel von Komponenten-Frameworks ist es einige Nachteile der objekt-orientierten Frameworks zu überwinden, wie zum Bei-

spiel die Probleme die bei der Komposition von objekt-orientierten Frameworks auftauchen und die eine Änderung an den Basisklassen der Frameworks notwendig machen. Im Gegensatz zu objekt-orientierten Frameworks fehlt derzeit jedoch noch ein geeignetes Vorgehensmodell für Komponenten-Frameworks. Um den Entwicklungsprozess von Komponenten-Frameworks zu verbessern, ist im Rahmen dieser Dissertation der ProMoCF- Ansatz (Abkürzung für "Process Model for Component Frameworks") entwickelt worden. Hierbei handelt es sich um ein leichtgewichtiges Vorgehensmodell und eine Entwicklungsmethodik für Komponenten-Frameworks. Das Vorgehensmodell wurde im gegenseitigen Nutzen mit der Entwicklung des MM4U-Frameworks erstellt. Während der ProMoCF-Ansatz evaluiert wurde, indem er zur Entwicklung des MM4U-Frameworks angewendet wurde, hat die Entwicklung des MM4U-Frameworks wiederum wichtige Rückmeldungen zur Verbesserung und Reifung des ProMoCF-Ansatzes geliefert. Diese wechselseitige Beeinflussung zwischen dem ProMoCF-Ansatz und dem MM4U-Framework führte zu einer ständigen Verbesserung des ProMoCF-Ansatzes und des MM4U-Frameworks über mehrere Iterationen hinweg. Zur Evaluierung des ProMoCF-Ansatzes ist dieser nicht nur zu Entwicklung des MM4U-Frameworks verwendet worden. Zudem ist auch das Ergebnis der Anwendung des ProMoCF-Ansatzes, also das MM4U-Framework, zur Entwicklung von mehreren Anwendungen in der Domäne der Multimedia-Personalisierung eingesetzt und evaluiert worden.

Das MM4U-Framework bietet eine generische Unterstützung zur dynamischen Erstellung von personalisierten Multimedia-Inhalten. Es stellt keine Neuerfindung der Adaption von Multimedia-Inhalten dar, sondern zielt auf die Vereinigung und Einbettung existierender Forschungsansätze und Lösungen im Umfeld der Multimedia-Personalisierung. Zudem kann das MM4U-Framework durch domänen- und anwendungsspezifischen Lösungen erweitert werden. Mit so einem Framework an der Hand können Anwendungsentwickler effizient eine dynamische Erstellung ihrer personalisierten Multimedia-Inhalte realisieren, die den Anforderungen der Benutzer entsprechen und damit die *letzte Meile* der Übermittlung von Multimedia-Präsentationen überwinden. Das MM4U-Framework wurde zur Entwicklung von mehreren Anwendungen aus unterschiedlichen Domänen angewendet. Diese Anwendungen, wie zum Beispiel ein personalisierter, mobiler, multimedialer Stadtführer, ein personalisierter, multimedialer Sportnachrichtenticker und ein personalisierter Photoalbumgenerator, zeigen die Anwendbarkeit des Frameworks.

Short Table of Contents

Contents

1 Introduction

Lost in Information Space, the human capacity to absorb information is almost saturated. Today, there is too much information offered at the same time, so much information that is not suitable for the current situation of the user, and too much time is needed to find information that is really helpful. However, access to proper information resources is essential for a competitive position in global society [Ams02a, Ams02b]. The multimedia material is there, but the issue how the multimedia content is found, selected, and assembled such that it is most suitable for the user's interest and background, the user's preferred device, network connection, location, and many other settings too is far away from being solved. Technologies and approaches which are needed here range from multimedia data management, building user profiles, modeling of multimedia meta data, intelligent multimedia information retrieval and analysis, multimedia document modeling, and multimedia content adaptation and composition to a personalized delivery and rendering of multimedia information on an end user's device [Ang03]. Consequently, with this thesis the challenge is tackled how today's multimedia applications can reach the goal to provide the right information to the individual user at the right time [SC00]. For it, we need to consider how the multimedia content can be personalized to the needs and requirements to the individual user in an efficient fashion.

Although personalization has become hype [RSG01] and the science behind personalization has undergone tremendous changes in recent years [MAB00], there is a clear awareness for personalized content and products and proper technologies are evolving and developing [Bol02]. The basic goal of personalized applications is to provide users with what they want or need without requiring them to ask for it explicitly [MAB00]. However, the development of personalized multimedia applications is complicated, as the creation of personalized content is an extensive process and involves different phases and tasks [Ang03]. To provide application developers a comprehensive and at the same time application-independent and practical support for the development of personalized multimedia applications, a proper means needs to be provided. Today, there are different systems and solution approaches for multimedia personalization. Though, these systems and approaches find their limits in regard of required effort, applicability and practicability, as well as their power of expressiveness. Consequently, we argue in this thesis for the need of a novel software engineering approach. We pursue the development of a component framework for personalized multimedia applications, called the MM4U (short for "MultiMedia For You") framework. The MM4U framework can be directly applied for an industrial like development of personalized multimedia applications. By this, it simplifies and improves the development process of personalized multimedia applications.

1.1 Background and Motivation

To illustrate the personalization of multimedia content, Figures 1.1 and 1.2 show a schematic depiction of a personalized city guide application[1] for our home town Oldenburg[2] in Germany. Figure 1.1 depicts an instance of the personalized tourist guide for Mr. Maier. He is interested in art and culture, i. e., likes to see and visit churches, museums, and theaters. The single spots on the city map indicate the sights selected according to his interests. The double encircled spot shows his current location. As he is currently in front of the palace of Oldenburg, he receives a presentation about this sight. The presentation includes some pictures of the sight and a textual description, according to Mr. Maier's preferences. As Mr. Maier is German, he receives the descriptions in his mother tongue.

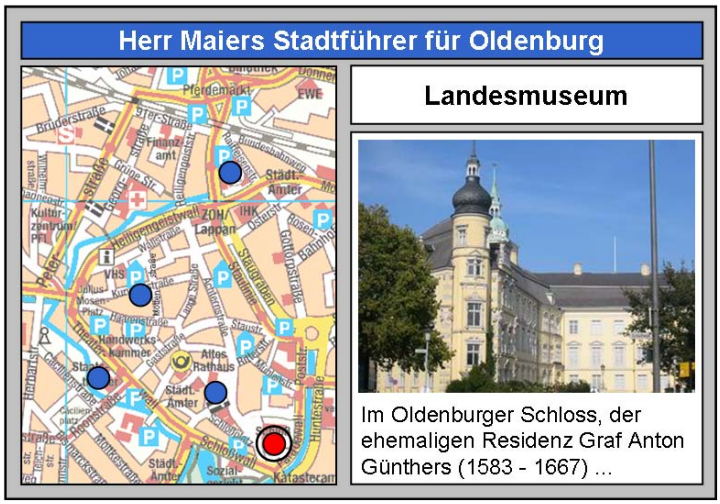


Figure 1.1: Personalized city guide application for Mr. Maier

Figure 1.2 shows the same personalized tourist guide application, however adapted to the needs of Mrs. Miller, a business women from Great Britain visiting Oldenburg. Mrs. Miller is searching for a good restaurant in Oldenburg. Thus, nearby restaurants are recommended by the application, such as the one depicted in Figure 1.2. Mrs. Miller prefers an auditive presentation of the spots matching her profile. As she is Briton, she receives her presentations in English.

The small example of a personalized city guide for Oldenburg already shows that with personalization of multimedia content a multitude of variation possibilities are involved. The variation possibilities for our personalized city guide are depicted in Figure 1.3. The root of the tree represents the multimedia presentation for the personalized city tour. Assuming that this presentation was intended for both Tablet PCs and Pocket PCs, this results in two variants of the presentation. If then some tourists are interested only in churches, museums, or palaces and would

[1] The city map image is used with permission from CCV GmbH, Germany.
[2] http://www.oldenburg.de/ (Received March 14, 2006)

Figure 1.2: Personalized city guide application for Mrs. Miller

like to receive the content in either English or German this already sums up to twelve variants. If then the multimedia content should be available in, e. g., different modalities, targeted at different (mobile) end devices, and delivered in different presentation formats, the number of variation possibilities within a personalized city tour increases again and again. Each path through the tree of variation possibilities depicted in Figure 1.3 shows one concrete instance of personalizing the multimedia content for a specific user. Assuming that Mr. Maier uses a Tablet PC for his city tour, the path as depicted in the left hand part of the Figure 1.3 could be valid for him. However, for taking the needs of Mrs. Miller into account, the path as shown in the right hand part is applied.

Even though different variants are not necessarily entirely different and may have overlapping content, the example is intended to illustrate that the flexibility of multimedia content to personalize to different users quickly leads to an exponential increase, i. e., an explosion of different options. And still the content can only be personalized within the flexibility range that has been anchored in the content. However, the manual creation of such personalized multimedia content is not practical not to mention economical due to the huge variety of different variation possibilities involved [AMR96b]. Hence, a dynamic creation of personalized multimedia content needs to be provided [SRM97].

The reasons why applications should provide for personalized information are manifold. Generally speaking, human beings like to be individually treated, considered, advised, and entertained. However, in places, where no personal contact is available, e. g., in the web or in mobile situations, systems have to replace the task of adequately offering individual communication and information. Thus, personalization can be seen as a current trend in information society [ELVKB01] and is one of the defining concepts of a contemporary multimedia platform [LNK04].

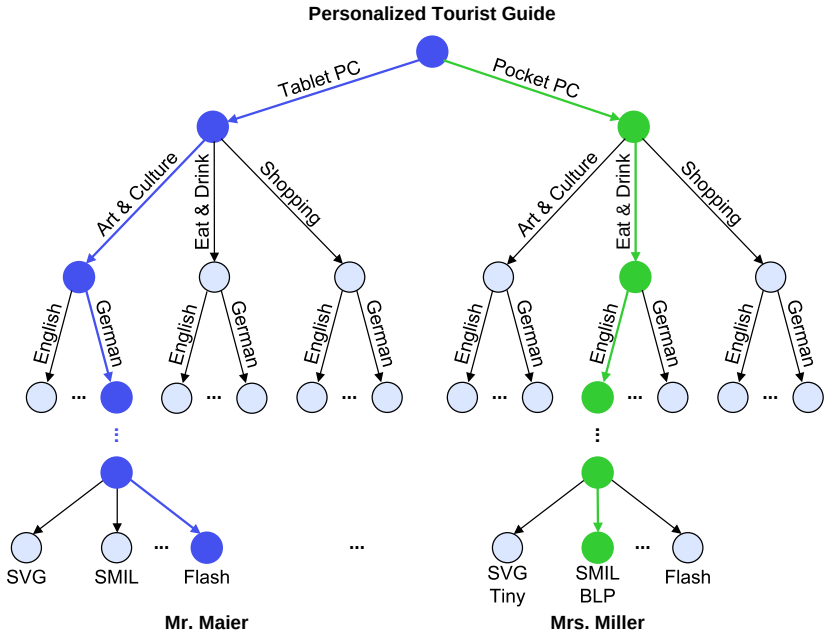


Figure 1.3: Example of the variation possibilities within a personalized city guide application

Personalization has demonstrated its benefits for both the users and the providers of personalized services [Kob01]. Nevertheless, the dynamic creation of personalized multimedia content and the development of applications providing such content requires high effort. Consequently, research on the dynamic creation of personalized multimedia content has been conducted by different research groups worldwide, e. g., at the research institute CWI in the Netherlands (see http://www.cwi.nl/) and at the INRIA Rhône-Alpes in France (see http://www.inrialpes.fr/). After a decade of research on this issue, we are pleased to observe today that there exist many scientific approaches as well as industrial solutions that provide personalized content to the user. The existing systems and approaches typically use declarative languages, e. g., rules and constraints, or exploit style sheets, transformation sheets, templates, and similar, to express the dynamic, personalized multimedia content creation. However, today's systems in regard of dynamically creating and providing personalized content are mainly text-centric. In addition, we observe that the existing approaches find their limits in regard of required effort, applicability and practicability, as well as their power of expressiveness. Whenever a complex and application-specific personalization generation task is required, the systems find their limit and need additional programming to solve the problem. Many of these systems and approaches for content personalization and adaptation are solutions for a specific application domain in which they provide a very specific content personalization task and are therefore limited to this very specific application area only. Consequently, a generic support for the dynamic

creation of personalized multimedia content and an efficient development of personalized multimedia applications is needed (cf. [Kob01]). The question is, however, how can multimedia applications solve the tough and expectedly expensive task of creating and delivering multimedia content that meets all the different individual user's settings?

With the proposed MM4U framework [SB05c, SB05d, SB04b], we pursue a software engineering approach for the creation of personalized multimedia content. The MM4U framework provides generic support for the development of personalized multimedia applications. Even though a software engineering approach towards dynamic creation of personalized multimedia content may not be the obvious one, we are convinced that our framework fills the gap between dynamic personalization support based on abstract data models, constraints or rules, and application-specific programming of personalized multimedia applications.

1.2 Goal of this Work and Proposed Solution

The aim of the MM4U component framework is to provide application developers with an extensive and application-domain independent support for the dynamic authoring of personalized multimedia content. The framework provides generic functionality for typical tasks of the general process chain of creating personalized multimedia content. This process chain starts with the selection of media elements such as audio, video, image, and text according to the user's individual profile and context information. These media elements are then assembled and composed in time and space using an abstract multimedia content representation model. This model captures the different aspects of the multimedia presentation such as the temporal course, the spatial layout, and the interaction possibilities without instantiating these in a concrete syntax and format. Only in the subsequent transformation phase—called the *last mile*—the multimedia content is transformed into the concrete multimedia presentation formats such as the standardized Synchronized Multimedia Integration Language (SMIL) [ABC+01b], SMIL in the Basic Language Profile (BLP) for mobile devices [ABC+01b], Scalable Vector Graphics (SVG) [AAA+04b], Mobile SVG [AAA+04a], as well as the industry-standard Flash [Mac04]. Finally, the multimedia presentation is delivered to the targeted (mobile) end device. Employing standardized multimedia presentation formats allows for employing existing multimedia player software on the end user clients. Thus, application developers are alleviated from the burden of developing their own multimedia playback engine.

To reach the goal of a software framework providing generic support for personalized multimedia applications raises the question of a proper software engineering support to develop such frameworks. A software framework, like the developed MM4U framework, typically constitutes a semi-finished software architecture for a specific domain. Since the introduction of object-oriented frameworks in the late eighties, the development of software frameworks is still costly and difficult to handle. To reduce development risks, process models and development methodologies have been developed. With the emergence of component technology also so-called component frameworks appeared. Component frameworks are aimed to overcome some drawbacks of object-oriented frameworks, such as problems that occur when

combining object-oriented frameworks and when changes at the bases classes of these frameworks become necessary. In contrast to object-oriented frameworks, however, a proper process model for component frameworks is still missing. Consequently, the quality of today's component frameworks is eventually dependent on the experience and skills of the framework developers. To improve the development process of component frameworks, a proper process model is needed. Such a process model needs support by a development method for identifying and specifying the framework's components. The goal of the dissertation here is the development of a lightweight process model and development method for component frameworks, called ProMoCF (short for "Process Model for Component Frameworks"). For evaluating the ProMoCF approach it has not only been applied for the development of the MM4U component framework. However, the outcome of the ProMoCF approach, i. e., the MM4U component framework itself, has been applied and evaluated for the development of several applications in the domain of multimedia personalization.

The MM4U component framework aims at providing generic support for the dynamic authoring of personalized multimedia content. It does not re-invent multimedia content adaptation but is targeted at incorporating and embedding existing research approaches in the field and allows to be extended by domain and application-specific solutions.

Summarizing, the goals of this work lie in contributing for the research areas of multimedia content engineering and software engineering. The goals are to provide generic support for the dynamic authoring of personalized multimedia content and to improve the development process of component frameworks.

1. Personalization of Multimedia Content:
   - ❏ Framework support for the development of personalized multimedia applications
   - ❏ Dynamic generation of multimedia content
   - ❏ Support for different end devices and standardized multimedia presentation formats
   - ❏ Integration of existing solution approaches and systems on different levels
   - ❏ Independence of the actual application domain
2. Improvement of the Development Process of Component Frameworks:
   - ❏ Systematic development of component frameworks
   - ❏ Method for identifying and specifying the framework components

1.3 Summary of the Contributions to the Scientific Community

As presented above, the research conducted in this work aims at the fields of multimedia content engineering and software engineering. Thus, the contribution of this work lies in both research areas. The benefit for multimedia content engineering is the MM4U component framework, providing generic support for the dynamic authoring of personalized multimedia content. For the software engineering research

community the dissertation contributes the lightweight ProMoCF approach improving the development process of component frameworks.

Benefits for Personalization of Multimedia Content Towards the creation of personalized multimedia content, we pursue a software engineering approach with the MM4U framework. The framework does not re-invent personalized multimedia content creation but is targeted at embracing and integrating existing approaches and solutions in the field and allows to be extended by domain and application-specific solutions. So far, an integration approach like the MM4U component framework has not been conducted. Though, the benefits are very promising as the different existing approaches and solutions for multimedia personalization can be employed within the framework. As the single approaches for multimedia content personalization have their individual advantages and disadvantages, we can exploit the advantages of each approach without necessarily having also to deal with their disadvantages. In addition, since the framework is prepared at integrating future approaches and solutions in the field, it is also of lasting value. New approaches can be integrated and thus the framework can be kept up to date to the most current technologies for multimedia content personalization.

For creating the personalized multimedia content, the MM4U framework provides a multimedia content representation model that is independent of the different presentation formats we find today. This model abstracts from the central aspects of multimedia content, i. e., the temporal course, spatial layout, and interaction possibilities of the presentation with the user [SB05c]. It is designed to be easily transformable to the different concrete presentation formats. For it, an abstract and application-independent transformation algorithm is supplied by the MM4U framework that provides for transforming the content in the abstract content representation model to the features and syntax of the concrete presentation formats [SB05d].

However, the MM4U framework also provides an abstract architecture for the domain of personalized multimedia applications. With its distinct interfaces, the different approaches and solutions in the field can be integrated. Developers of concrete applications basing on this framework do not need to design their own application architecture but can rely on the pre-fabricated, semi-finished architecture the MM4U framework provides and adapt it to their own needs. With such a framework at hand, application developers can efficiently realize on-the-fly multi-channel generation of personalized multimedia content that meets the end user's requirements and overcome the *last mile* in multimedia presentation delivery. Employing standardized multimedia presentation formats allows personalized multimedia applications to use and rely on existing multimedia players on the market. This is especially important in the heterogeneous mobile world, where we find new mobile devices such as cell phones and PDAs (Personal Digital Assistants) every six month.

Improvement of the Development Process of Component Frameworks To reach the goal of a generic support for developing personalized multimedia applications by means of the MM4U framework, the question rises how such a component framework can be developed. As currently there exists no systematic support for the development of component frameworks, the development of such frameworks solely

depends on the skills and experience of the framework developers. The dissertation contributes the ProMoCF approach to improve the development process of component frameworks. The lightweight process model and development method bases on the hot-spot-driven approach for object-oriented frameworks developed by Wolfgang Pree [FPR02, Pre99].

ProMoCF extends the hot-spot-driven approach for object-oriented frameworks by activities and methodical support for identifying the framework's components and defining the framework's flexibility requirements by introducing so-called group-hot-spot-cards. Although the ProMoCF approach is not an all embracing process model solving all problems and challenges of developing and evolving component frameworks, it is a very valuable improvement towards a systematic software engineering support for component frameworks.

1.4 Overview

The following Figure 1.4 shows the structure of this work. Starting with the goal to provide a generic support for efficiently developing personalized multimedia applications with the MM4U component framework, two research traces are pursued. The left hand trace considers the challenge of personalizing multimedia content from the multimedia community point of view. It encompasses the conceptual design of the MM4U framework. In contrast, the right hand trace pursues the improvement of the development process of the component framework. It contains the development of our process model and a development method for component frameworks ProMoCF. Finally, both traces converge and result in the development and application of the MM4U component framework.

The dissertation is structured in three larger logical parts. These are the fundamentals and related work, the conceptual design of the MM4U framework and the design of the ProMoCF approach, as well as the development and evaluation of the MM4U framework and ProMoCF approach. As depicted in Figure 1.4, each part of this work consists of different blocks. Each block represents a distinct step to conduct to reach the goal of this work. The directed arrows between the blocks show the dependencies and influences of these steps. The numbers in parentheses on the bottom right side of each block indicate the associated chapters of this dissertation.

Based on the structure provided by Figure 1.4, the remainder of this work is organized as follows: Subsequent to this introduction, three representative application scenarios of personalized multimedia applications are described in Section 2. To review the notion of multimedia content and multimedia authoring, in Section 3 we present the requirements of multimedia content modeling and the authoring support we find today. Setting off from this, the same section introduces the reader into the tasks of creating personalized multimedia content and why such content can be created only in a dynamic fashion. Here, we also address the related approaches we find in the field of multimedia personalization. In Section 4, we introduce the notion of software frameworks. We distinguish between two kinds of frameworks, the object-oriented frameworks and component frameworks. The development of software frameworks in general is presented and the current support for the development of object-oriented frameworks and component frameworks is described. As the ProMoCF approach for component frameworks bases on the hot-spot-driven

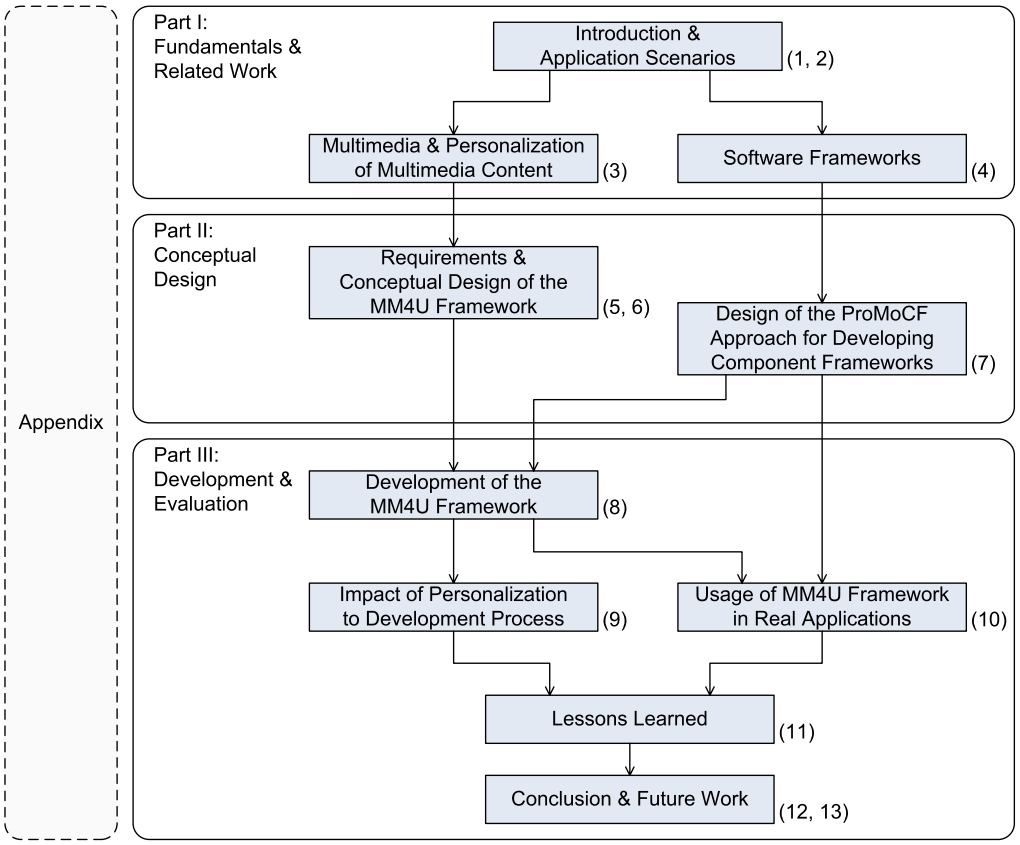


Figure 1.4: Overview of the structure of the work

approach for object-oriented frameworks by Wolfgang Pree, the hot-spot-driven approach is presented in detail.

In Section 5, the general process chain for personalizing multimedia content is introduced. In addition, the requirements to the MM4U approach are identified as well as a layered architecture of the MM4U framework is defined. In Section 6, we present the conceptional design and data structures underlying to the framework layers in detail. The design of our lightweight process model for component frameworks ProMoCF is described in Section 7. For it, we first argue for the benefits of a lightweight process model for component frameworks. Then, the design of our ProMoCF approach on basis of the hot-spot-driven approach is described.

The concrete development and implementation of the MM4U component framework by applying the ProMoCF approach is presented in Section 8. Here, the development of the framework is described along the single activities defined by the ProMoCF approach. The specification, design, and implementation of each framework component is described and the general usage and deployment of the framework is presented. In addition, also the definition of a mobile subset of the MM4U framework, called the mobileMM4U framework, is introduced. The mobileMM4U

framework constitutes a specific configuration of the MM4U framework's components specifically targeted at the development of mobile personalized multimedia applications. In Section 9, the impact of personalization to the multimedia software engineering process is presented. The MM4U component framework has been employed for the development of several personalized multimedia applications from different domains. These demonstrator applications, e. g., a personalized mobile multimedia city guide, a personalized multimedia sports news ticker, and a personalized picture albums generator, show the applicability of the framework and are presented in Section 10. The experiences and knowledge gained with developing the MM4U component framework by applying and evaluating the ProMoCF approach are summarized in Section 11. Finally, a summary of the thesis is given in Section 12, before we conclude this work with a view to future work.

2 Application Scenarios

In this section, three illustrative application areas in the domain of personalized multimedia applications are introduced and analyzed. Each application area has its individual focal point in regard of the issues of multimedia personalization. Together, they cover a broad spectrum of applications in the domain of multimedia personalization. The three application areas are:

- Mobile multimedia applications: The first application area deals with personalized, mobile, multimedia information systems. Within this application area, systems such as personalized mobile tourist guides and mobile games are considered.
- Multimedia information services: The second application area considers personalized, multimedia services and presentations. This area comprises among other things personalized multimedia advertisement and personalized multimedia news.
- E-learning applications: Within the third application area of personalized, multimedia, e-learning systems, approaches like adaptive hypermedia systems [WKDB01, DBBH99, Bru96] and intelligent tutoring systems [Sch97] are considered.

The three application areas as sketched above are presented in detail in the following Sections 2.1 to 2.3. For it, a representative application scenario for each application area is described. Each application scenario has a chief character. The role and situation of this character is introduced first, before the actual application scenario is presented.

2.1 Mobile Multimedia Applications

- Role: a day in life of business woman Mrs. Miller.
- Situation: Mrs. Miller is attending a multimedia conference in Oldenburg, Germany.

The first application area of mobile multimedia applications deals with a regular day in life with the business woman Mrs. Miller from Great Britain. The business woman Mrs. Miller is attending the fictive multimedia conference MMCO in Oldenburg, Northern Germany.

Traveling to Oldenburg The business trip of Mrs. Miller begins already early in the morning by flight from London to Bremen. While traveling by train from Bremen

to Oldenburg, she reads her personalized multimedia news-magazine via WLAN in the train using her Tablet PC (see also Section 2.2) and catches up on the latest daily news. Finished with reading the news, Mrs. Miller visits the city center of Oldenburg in a virtual tour through the pedestrian-zone. According to Mrs. Miller's interests, the virtual tourist guide recommends a set of interesting sights. For example, the tourist guide draws her attention to the south of the city center. Here, the Schlossplatz and the palace of Oldenburg is located. Mrs. Miller requests further information about the building and learns that a museum for art and cultural history is located in the palace of Oldenburg. Since Mrs. Miller is interested in art from the antiquity, she receives a message that currently an exhibition about art in the antiquity takes place in the state museum. As Mrs. Miller is planning to take a tour through Oldenburg's city center in the evening, she decides to add a visit of the exhibition into her current trip-plan.

Around twelve o'clock the train arrives at the main station in Oldenburg, early enough to reach the auditorium of the university by bus, where the multimedia conference is located. After attending some interesting presentations at the conference, she gets tired. So, Mrs. Miller relaxes by continuing the preparation of her tour through the city center in the evening, using her PDA this time. Of course the other listeners shall not be disturbed by it. Therefore, all auditory information is replaced by additional visualizations and textual information. When she finished her tour preparation and the first day of the conference ends, Mrs. Miller goes into her hotel in the city center.

The Personalized City Tour in the Evening After a short rest in the hotel and a nice dinner, Mrs. Miller finally starts her city tour through Oldenburg, which she has prepared in the train and at the conference. Her personalized mobile assistant proposes a route through the city center and also takes the restricted opening hours of the exhibition in the state museum into account. On the city map on her mobile assistant, the individual sights of the city tour are indicated, like for example the Lamberti church and the palace at the Schlossplatz. In addition, also further sights are presented on the map, although they are not part of her current tour. However, these sights match the interests profile of Mrs. Miller and can be added to the current tour at any time.

Accompanied by her mobile assistant Mrs. Miller now explores the city center of Oldenburg. Thereby, she gets further multimedia information about the sights on request. As Mrs. Miller is Briton, her preferred language is English and thus receives the presentations in her mother tongue. Despite the already late evening hour, the sun still glares quite strongly over Oldenburg at that sunny summer day such that Mrs. Miller can't read the display of her mobile assistant very well. Therefore, she toggles the city tour application to purely acoustic output. During her tour, she receives presentations with further information about the sights she is visiting.

Arriving at the palace, the mobile assistant asks Mrs. Miller whether she would like to visit the exhibition in the state museum now. Since there is enough time left, Mrs. Miller approves and is automatically referred to the personalized exhibition guide of the museum. However, so much culture is very exhausting. Mrs. Miller therefore asks her mobile assistant for a nearby located, quiet pub where she—according to her personal preference—can drink a beer before she goes back to the hotel. The mobile assistant proposes Mrs. Miller the nearby located cellar of the old

town hall (similar to Figure 1.2). Mrs. Miller watches a personalized multimedia presentation about the cellar, which includes a short video clip about the pub as well as the current beer prices. She decides to have a glass of beer there, before she returns to her hotel.

2.2 Multimedia Information Services

- Role: an ordinary morning with family Maier having their breakfast at their home in Munich.
- Situation: Mr. Maier reads his multimedia-based daily newspaper.

The second application area considers a normal day in life with family Maier in Munich. It is an ordinary Monday morning and family Maier is having their breakfast at their home.

Mr. Maier Reads His Multimedia-based Daily Newspaper A usual day with family Maier having their breakfast. While Mrs. Maier eats her jelly-gem and her son stirs listlessly in his cocoa, Mr. Maier catches up on the latest news reading his daily multimedia newspaper on his laptop (see also Section 2.1). This newspaper is personalized in regard of his individual interests. This means that besides the main headlines and world news, the newspaper contains in particular those news directly on the title page that are important and interesting for Mr. Maier. His attention is attracted by the stock prices and the business news. Further below, an animation shows the current weather in Munich.

After Mr. Maier gained an overview about his personal news, he opens the article about the current situation in Middle East. The article is dynamically generated according to the user profile information of Mr. Maier. While a video within a multimedia presentation shows that the crisis becomes worse, further facts and information about the topic are presented from time to time, just like Mr. Maier's preferences determine it. His son, politically rather uninterested, but very familiar with his cell phone, has to prepare a presentation about Middle East for school. Therefore, Mr. Maier forwards the article to the cell phone of his son. Here however, the layout of the presentation is adapted to the cell phone's limited display size and network connection. In addition, the article's content is presented in a variant which is more suitable for children. Since it has become already quite late in the morning, Mr. Maier finishes his breakfast quickly and sets out for work.

Mr. Maier Receives a Personalized Product Advertisement At work, Mr. Maier receives an e-mail from the car manufacturer he bought his current car some years ago. As Mr. Maier is thinking about purchasing a new car for a while now, he decides to open and read the e-mail during his lunch break. When opening the advertisement, a personalized multimedia presentation for the recent sports car starts.[1]

[1] Before it starts, the technical characteristics of the client device's hardware and software are send to the server. That are among other things the name and the version of the applied multimedia player, the operating system, the speed of the Internet connection, the computing power of the processor, the displays characteristics as well as the size and type of the

Varnished in Mr. Maier's favorite color and seat covering the powerful sports car drives in a video clip through a mountainous scenery, just like the place where Mr. Maier lives. While the video is playing, Mr. Maier can interactively retrieve further information about the car, e. g., the offered space, consumption, and handling characteristics. While exploring the car's characteristics, animations are exploited to explain the selected item in a better way. In addition, he can also interactively change the characteristics of the car, such as the interior of the vehicle, color, and other technical specifications. Mr. Maier notices that the advertised car virtually corresponds to what he has told his car dealer when he bought his current car in regard of color, equipment, and price. Stimulated by the appealing product presentation, Mr. Maier arranges a test drive with his car dealer.

Mr. and Mrs. Maier Watch Their Favorite Soap-opera After an exhausting though successful working day, Mr. Maier and his wife drink a glass of wine and watch their favorite soap-opera on digital TV to let the day fade away. At a thrilling scene, the soap is interrupted by an advert break of course. Since Mr. and Mrs. Maier have no longer subscribed the *bronze television program* but changed to the more expensive *silver subscription*, the advert break is only half the time than it used to be before. As they voluntary stated some personal information to the broadcaster, e. g., their hobbies, type of music and literature they like, number of children, and which products they buy in the supermarket, the additional expenses for the *silver subscription* are similar to their previous *bronze subscription*.[2] Consequently, the commercials are selected according to Mr. and Mrs. Maier's interests and preferences. By this, they avoid to get bored with uninteresting and inappropriate advertisement clips while saving at the same time money for the upgraded subscription. For example, for Mr. Maier a commercial about the new sports car is presented like he has seen in the e-mail advertisement during his lunch break. For Mrs. Maier a commercial is selected presenting the new Madonna album. Since it has already passed 8 p.m. in Munich and Mr. and Mrs. Maier like to drink beer also a commercial about a new flavored beer is added to the advert break. When the presentation of the advertisement break is finished, the TV program returns to the regular program and Mr. and Mrs. Maier continue watching their soap-opera.

2.3 E-learning Applications

- Role: an hour in the life of a pupil at elementary school and grammar school.
- Situation: Learning the mode of action of antibiotics with help of an adaptive e-learning system.

The third application area deals with an hour in life of a pupil at elementary school and grammar school. In this application scenario the pupils are learning the mode of action of antibiotics with the help of an adaptive e-learning system.

storage medium. Since Mr. Maier has a powerful multimedia computer and a broadband Internet connection, a high-resolution video is selected together with many further multimedia supplements.

[2] The most expensive *gold subscription* would be without any advert breaks.

Pupil at Elementary School Peter, a pupil at the third class of an elementary school, has been given the e-learning system *My body* as a birthday present. At the start-up screen of the application, a comic-like, virtual assistant asks Peter for his first name. By this, Peter is identified by the e-learning system. The virtual assistant greets Peter and leads him through the content of *My body* in a so-called guided tour. Peter is currently ill and thus his family doctor prescribed him some antibiotics. Although, Peter does not like to take the pills, he is interested in getting some information about *What is when I feel sick?* by his e-learning program.

When Peter clicks on this section of his e-learning program, the virtual assistant pops-up and asks him whether he is ill. The assistant consoles Peter with some warm words and leads him to a chapter about antibiotics. Here, Peter experiences in a guided tour by means of a rich multimedia presentation in a playful and for children suitable way why he needs to take antibiotics to become healthy. First, it is explained what bacteria are and how they can intrude into his body. Then, Peter gets to know what bacteria do in his body and that these multiply there. Finally, the mode of action of antibiotics is explained how they help him to recover.

Pupil at Grammar School At a biology lesson at a grammar school, the pupils learn about the *Functionality of antibiotics* by employing the adaptive e-learning system called *The human being*. As the other pupils, also Benny logs into the e-learning system by his login and password. The degree of difficulty, i. e., the level of twelfth class, as well as the stereotype *pupil at grammar school* was selected by the pupils' teacher. Moreover, some settings of the e-learning system can be done by the pupils themselves, e. g., the preferred presentation style and the character of the virtual tutor they like most.

After successful log in, Bennie gets a multimedia questionnaire targeted at pupils level to investigate his prior knowledge about the functionality of the human body and antibiotics in particular. Benny answers the first questions successfully, but has some problems on the question *What are antibiotics?*. The system does not ask him any further questions about the functionality of antibiotics, but continues with some other ones.

Finished with the grading test, Benny selects the chapter *Functionality of antibiotics* in the e-learning system's navigation menu. Thereupon, a multimedia e-learning unit about antibiotics is generated. This e-learning unit is personalized to the prior knowledge of Benny. With the multimedia e-learning unit, Benny learns about antibiotics and their mode of action. When finished with the unit, some questions have to be answered by Benny. These questions are inserted by Benny's teacher into the e-learning unit and need to be successfully answered to continue with the learning process. If Benny answers almost all questions of a e-learning unit correctly, he can proceed with other e-learning units. The e-learning units are more or less interdependent, i. e., certain units rely on other ones building a dependency graph. With the adaptive e-learning system, only those units of the dependency graph can be learned by Benny he has the necessary prior knowledge acquired. If Benny has problems with answering the questions of an e-learning unit, he can ask his personal virtual tutor integrated in the e-learning system for some advice. The tutor helps Benny with answering the questions and proposes the most suitable e-learning units for review.

The same e-learning system is also available for students of medicine. In contrast to the pupils of a grammar school, however, this system provides arbitrary access to the e-learning units, i. e., all units and sections can be accessed anytime, serving the principle of self-determined adult learning.

2.4 Observations from the Application Scenarios

The three application scenarios introduce the subject matter of personalization. They were selected in order to cover a broad spectrum of personalized multimedia applications. Consequently, they consider a comprehensive amount of personalization features and comprise different aspects of personalization. Although these aspects and features are intersecting, one can clearly observe that each area has its own focal point and requirements in regard of multimedia personalization. In the following Sections 2.4.1 to 2.4.3, the observations from the three application scenarios are presented as well as the aspects of personalization involved are described. In Section 2.4.4, some final conclusions from these observations are drawn.

2.4.1 Mobile Multimedia Applications

The most important personalization aspect that mobile multimedia applications take into account are the location and information about the user's surrounding and environment, such as weather and noise. This is often subsumed under the notion of context. Here, typically location-based information or location-based services are provided [Sne01, Sea06], such as the personalized tourist guide of Mrs. Miller presented in Section 2.1 presents information about sights she is standing in front of or gets recommendations about nearby located pubs.

Other personalization aspects that are taken into account are information about the users' interests and preferences, which are typically exploited by mobile tourist applications. With the area of mobile applications, also a change of modality needs to be taken into account. For example, if the lighting is too high to read the display, only auditive information is possible (Mrs. Miller switches her tourist guide to auditory information) or if the surrounding requires it, only visual information must be provided (Mrs. Miller finishes the preparation of the sightseeing tour during a conference presentation). Changing modality is also important in the mobile context to reduce the workload of the user. For example, it is difficult to read a text or watching a video on a mobile device while walking through a crowded pedestrian zone.

However, another important challenge with mobile multimedia applications is to deal with the huge variety of different mobile end devices such as cell phones and PDAs. Serving this heterogeneous market of mobile devices is very difficult due to the varying technical characteristics and software settings of the devices. This is even amplified if personalized applications need to serve not only mobile devices but also stationary systems such as Desktop PCs (like Mrs. Millers tourist guide application is used on her notebook for pre-trip planning and on her PDA for the actual tour). Here, the content needs to be adapted to the different (mobile) end devices, taking their different technical characteristics into account.

2.4.2 Multimedia Information Services

With multimedia information services, typically information and assumptions about the user's interests, preferences, habits, buying history, and the like are exploited. The aim here is to provide personalized information such as news and advertisement (for instance the car advertisement for Mr. Maier in Section 2.2) or to recommend new services and products (like the personalized advertisement break for Mr. and Mrs. Maier). For the latter also typical context information needs to be taken into account. For example, commercials for alcohol and cigarettes on TV (if allowed at all) will be shown in the evening rather than in the morning.

The main challenge with this application area is to convince the user that the benefits of the information and services provided by a personalized system are high enough to divulge the necessary information about his or her personal habits and preferences. Personalized multimedia information services are only accepted if benefits for the user are high enough, i. e., if the user gets important information at the right time and if he gets benefits from using it (cf. [PR93]).

2.4.3 E-learning Applications

In contrast to the other application areas, the area of personalized e-learning applications focuses on modeling the cognitive aspects of the user. The cognitive aspects are traditionally considered by the research area of user modeling, which is going on for some decades now. Although the user's preferences and interests exploited by the other application areas are also subsumed under the notion of user modeling [FKS97b], the area of e-learning considers further user modeling aspects such as goals and plans, abilities and disabilities, and in particular the knowledge of the user. Here, the challenge is how to convey the same information for different users and user groups by personalizing it in regard of their different knowledge levels, point of views, didactic model proper to learning habits, and prior knowledge.

Taking different knowledge levels into account means to convey information to different target groups within their individual degrees of difficulty such as beginner, intermediate, and advanced. The different target groups of the e-learning system presented in Section 2.3 are the pupils from elementary school and grammar school as well as the students at university. With the presented e-learning system, first the prior knowledge of the pupils is acquired. This is a complicated and difficult task [KG02, ABDG02, Rhe03]. On basis of an initial model about the prior knowledge (e. g., using a stereotype) an adaptive e-learning system can constantly adapt and improve its information and assumptions about the users' knowledge [XZM02].

In addition, the e-learning content possibly also needs to be presented from different point of views on the domain, i. e., it needs to be personalized in regard of profession, role, or education. For example, the pupil Benny at the grammar school needs rather common information about antibiotics while, e. g., a veterinary student at university is probably more interested in specific information taking the envisioned profession into account. Another example where different point of views need to be provided is to present the topic of antibiotics to the specific needs of a, e. g., medical doctor, hospital nurse, or patient.

While acquiring the e-learning content, different users have different learning habits and needs. Taking this into account, didactic models define how to orga-

nize and convey the e-learning content. The didactic concept is reflected, e. g., in the structure of the e-learning content and navigation structure. For example, with our e-learning system a pupil at elementary school receives a guided tour with only limited navigation possibilities, while a pupil at grammar school can select some e-learning units. However, the access to the e-learning units is restricted to those the pupils have already gained the necessary knowledge. With the student at a university, arbitrary access to the complete content of the e-learning system is provided.

2.4.4 Conclusion

From the application scenarios described above and their reflective analysis here, we conclude that an efficient and economic support for developing personalized multimedia applications necessarily needs to provide domain-independent support for creating personalized multimedia content. This means, it needs to be independent of the concrete application domain, such as mobile tourism, sports news, and e-learning. Consequently, we need a generic support for creating personalized multimedia content that can be applied by application developers to develop their concrete application in their specific domain. In order to provide such generic support for the most different application domains, we need to provide support for the most different personalization aspects and features in these domains. This includes not only the technical characteristics of the used end devices, the user's current location and environment, but also the user's individual preferences, interests, and needs. To provide support for these different personalization aspects and profile information a concrete user can have, an abstract user model is needed. This abstract user model allows to manage the user profile information in the different domains. From the application scenarios, we can further derive that the envisioned generic support for personalized multimedia applications must not only provide for authoring personalized content in the most different domains. It must also provide support to author personalized content for different end devices, including Desktop PCs, PDAs, and cell phones.

3 Authoring of Personalized Multimedia Content

> The only man who behaves sensibly is my tailor; he takes my measure anew every time he sees me, whilst all the rest go on with their old measurements, and expect them to fit me.
>
> George Bernard Shaw (1856-1950),
> Irish-born British playwright

Having introduced in the previous section representative application scenarios for personalized multimedia applications, this section now introduces the reader in the notions and issues of multimedia content modeling, multimedia content authoring, personalization, and personalized multimedia content authoring in Section 3.1. In Section 3.2, todays support for the dynamic authoring of personalized multimedia content is analyzed. A systematic categorization of this authoring support for multimedia personalization is conducted in Section 3.3. Finally, we draw some conclusions based on this analysis and categorization for the MM4U approach in Section 3.4.

3.1 Notions and Issues of Personalizing Multimedia Content

An understanding of requirements and characteristics for modeling and authoring multimedia content is a helpful prerequisite to our goal, the dynamic creation of personalized multimedia content. Therefore, we first present our notion of media and multimedia in Section 3.1.1. In the following Section 3.1.2, the notion of multimedia content, multimedia document models, and multimedia presentation formats are introduced. In addition, the central modeling characteristics of multimedia content are identified and presented. For the creation of multimedia content, the notion of multimedia content authoring in is introduced in Section 3.1.3 and the manual mul-

timedia content authoring process is briefly described. In Section 3.1.4, our notion of personalization and the different aspects that are involved with it are presented. On basis of the definitions of multimedia content, multimedia content authoring, and personalization, the notion of personalized multimedia content is introduced in Section 3.1.5.

3.1.1 Media and Multimedia

With the increasing performance of modern computers and the support for a high quality playback of images, audio, and video in the beginning of the nineties, multimedia has entered numerous application domains. Today, we experience that multimedia is reaching mobile end devices, such as PDAs and cell phones. For the notion of multimedia, there exist many different definitions. Basically, multimedia is considered as a composed word consisting of "multi" (multiple) and "media" [SN95, Kje91]. This means that multimedia is the combination and usage of multiple media [Mer06a]. A medium can be either discrete or continuous. This is determined by its medium type (or medium form [HMHGK01, Koe94b]). Examples of media types for discrete media are text and image, e. g., computer graphics and pictures taken from a digital camera. Media types of continuous media are audio (e. g., sound and voice), video, and animation (cf. [Enc05]). An instance of such a medium type is called a medium element. Examples for continuous media elements are instances of the media types video, audio, and animation, while instances of the image type and text type constitute discrete media elements. Continuous media elements are also called time-variant media elements [GBE96], temporal media elements [GT95], or time-dependent media elements [BKCD98, SN95]. Discrete media elements are also known as time-invariant media elements [GBE96], non-temporal media elements [GT95], or time-independent media elements [BKCD98, SN95], respectively.

A concrete medium element can be of different media formats. A medium format is a defined data structure for a specific medium element. This data structure encodes the medium element. Typical media formats we find today for image elements are PNG [W3C03], JPG (or JPEG) [CCI93, W3C06b], and GIF [Com90]. Text elements are typically encoded by using ASCII characters. For encoding audio elements, typically media formats such as Fraunhofer IIS' MP3 [Fra06] or IBM's and Microsoft's WAV [Wik06] are employed. Typical examples of media formats storing a video element are AVI [Mic06b], QuickTime movie (MOV) [App06], and MPEG (MPEG or MPG) [MPE06].

Based on the analysis above, multimedia is seen as an interactive conveyance of information that includes (a seamless integration of) at least two media elements of the presented media types [HMHGK01, Tan98]. This definition is extended by Steinmetz et al. [SN95, HS91] in regard of storage and communication aspects of multimedia and refined towards the media types that need to be combined at least. This leads to a definition of multimedia that is characterized by the computer-controlled, integrated creation, manipulation (i. e., interaction of the user with the media), presentation, storage, and communication of independent information. This independent (multimedia) information is encoded at least through one continuous and one discrete medium element.

3.1.2 Multimedia Content and Content Representation

Multimedia is defined in the previous section as conveying information encoded in the combination of at least one discrete and one continuous medium element. Consequently, multimedia content is seen as the result of a composition of different media elements such as images, text, audio, and video into an interactive continuous multimedia presentation, comprising at least one discrete and one continuous medium element. Features of such a composition are the temporal arrangement of the media elements in the course of the presentation, the layout of the presentation, and the definition of its interaction features. Multimedia content builds on the modeling and representation of the different media elements that form the building bricks of the composition.

Based on this definition of multimedia content, we introduce a multimedia document as representation of multimedia content (also called multimedia object [ABH97]). A multimedia document that is composed in advance to its rendering is called pre-orchestrated in contrast to compositions that take place just before rendering that are called live or on-the-fly. A multimedia document is an instantiation of a multimedia document model that provides the primitives to capture all aspects of a multimedia document. The power of the multimedia document model determines the degree of the multimedia functionality that documents following the model can provide. Representatives of (abstract) multimedia document models in research can be found with Madeus [JLR+98, JLSI97], Amsterdam Hypermedia Model [Har98, HRJM94], CMIF [BRL91], COMMAND [GBE96], and ZYX [BK01, Bol01].

A multimedia document format or multimedia presentation format determines the representation of a multimedia document for the document's exchange and rendering. Since every multimedia presentation format implicitly or explicitly follows a multimedia document model, it can also be seen as a proper mean to "serialize" the multimedia document's representation for the purpose of exchange. Multimedia presentation formats can either be standardized such as the World Wide Web Consortium (W3C) standards SMIL and SVG, the International Organization for Standardization (ISO) standard Lightweight Applications Scene Representation (LASeR) [ISO06, ISO05], or proprietary such as the wide spread Flash format.

A multimedia presentation is the rendering of a multimedia document. It comprises the continuous rendering of the document in the target environment, the (pre)loading of media data, realizing the temporal course, the temporal synchronization between continuous media streams, the adaptation to different or changing presentation conditions, and the interaction with the user.

In the related work, there is not necessarily a clear distinction between multimedia document models and multimedia presentation formats. However, in this work we distinguish the notion of multimedia document models and multimedia presentation formats. Here, a document model is an abstract definition of multimedia composition capabilities and characteristics. An instance of a document model is called a multimedia document. A presentation format then constitutes a serialization of such a document for the purpose of exchange.

Independent of the actual document model or presentation format chosen for the content, one can say that a multimedia content representation has to realize at least three central aspects. These central aspects are the temporal

course, spatial layout, and interaction possibilities of a multimedia presentation [BK01, BKW99b, IB05, HCV04, JB01, RBO+00, Koe94a]. However, as many of today's concrete multimedia presentation formats can be seen as representing both a document model and an exchange format for the final rendering of the document, we use these as an illustration of the central aspects of multimedia documents. We present an overview of these characteristics in the following; for a more detailed discussion on the characteristics of multimedia document models we refer the reader to [BK01, BKW99b].

- Temporal course: A temporal model describes the temporal order between media elements of a multimedia document [BKW99b, BKL96, JB01, Fuh94, LG90a]. With the temporal model, the temporal course such as the parallel presentation of two videos or the end of a video presentation on a mouse-click event can be described. One can find four types of temporal models: point-based temporal models, interval-based temporal models [LG93, All83], enhanced interval-based temporal models that can handle time intervals of unknown duration [DK95, HFK95, WR94], event-based temporal models, and script-based realization of temporal relations. The multimedia presentation formats we find today realize different temporal models, e. g., SMIL 1.0 [BBB+98] provides an interval-based temporal model only, while SMIL 2.0 [ABC+01b] also supports an event-based time model.
- Spatial layout: For a multimedia document not only the temporal synchronization of these elements is of interest but also the spatial positioning on the presentation media, e. g., a window, and possibly the spatial relationship to other visual media elements [Ang03]. The positioning of visual media elements in the multimedia presentation can be expressed by the use of a spatial model. It defines the spatial organization, i. e., the spatial positioning of the visual elements [BKW99b, BKL96, JB01]. For example, one can place an image above a caption or define the overlapping of two visual media elements. Besides the arrangement of media elements in the presentation also the visual layout or design is defined in the presentation. This can range from a simple setting of background colors and fonts up to complex visual design and effects. In general, three approaches to spatial models can be distinguished: absolute positioning, directional relations [PTSE95, PS94], and topological relations [EF91]. With absolute positioning typically both the placement of a medium element at an absolute position with respect to the origin of the coordinate system as well as the placement of a medium element at a position relative to another multimedia composition element is subsumed. The absolute positioning of media elements with respect to the origin of the coordinate system can be found, e. g., with Flash [Mac04] and SMIL 2.0 in the Basic Language Profile (BLP) profile [ABC+01b], while relative positioning is realized, e. g., by SMIL 2.0 [ABC+01b] and SVG 1.2 [AAA+04b]. To make difference between the two possibilities explicit, we define in this work with absolute positioning the layouting of visual media elements with respect to the origin of the presentation's coordinate system. The placement of the visual media elements at positions relative to other multimedia composition elements is called relative positioning.

- Interaction possibilities: A very distinct feature of a multimedia document model is the ability to specify user interaction in order to let a user choose between different presentation paths [BKL96]. Multimedia documents without user interaction are not very interesting as the course of their presentation is exactly known in advance and, hence, could be recorded as a movie. With interaction models a user can, e. g., select or repeat parts of presentations, speed up a movie presentation, or change the visual appearance. For the modeling of user interaction, one can identify at least three basic types of interaction [BKL96]: navigational interactions, scaling interactions, and movie interactions. navigational interaction provides for control of the flow of a presentation. It allows the selection of one out of many presentation paths and is supported by all the considered multimedia document models and presentation formats (cf. hyperlink in [JB01]). Scaling interaction and movie interaction allow the user to interactively manipulating the visible and audible layout of a presentation [BK01, BKW99b]. For example, one could define if a user is allowed to change the presentation's volume or spatial dimension. However, scaling interaction and movie interaction are rarely used or not defined within today's multimedia presentation formats. Typically, such type of interaction rely on the functionality offered by the actual multimedia player used for playback of the presentation.

Looking at existing multimedia document models and presentation formats both in industry and research, one can see that these aspects of multimedia content are implemented in two general ways: The standardized formats and research models typically implement these aspects in different variants in a structured fashion as can be found with SMIL 2.0, HTML+TIME [SYS98], SVG 1.2, Madeus [JLR+98, JLSI97], and ZYX [BK01] employing the Extensible Markup Language (XML) [W3C04]. Proprietary approaches, however, represent or program these aspects in an adequate internal model such as the Flash format [Mac04].

For the actual rendering and playback of the multimedia presentation, so-called multimedia player software is employed. Today, multimedia player software is available for the most different devices, operating systems, and presentation formats. Examples of multimedia player software are the RealPlayer [Rea06], Ambulant Player [BJK+04, CWI05], Adobe SVG plug-in [Ado06a], Adobe's Flash plug-in [Ado06c] (formerly Macromedia), X-Smiles [X-S06], PocketSMIL [INR02], PocketSVG [CSI05], IBM's MPEG-4 player [IBM06a], and TinyLineSVG [Gir06] for Desktop PCs and mobile devices, respectively.

3.1.3 Authoring of Multimedia Content

While multimedia content represents the composition of different media elements into a coherent multimedia presentation, multimedia content authoring is the process in which this presentation is actually created. The process of multimedia content authoring involves parties from different fields including domain experts, media designers, and multimedia authors.

Experts from the domain provide their knowledge in the field; this knowledge forms the input for the creation of a storyboard for the intended presentation. Such a storyboard often forms the basis on which creators and directors plan the imple-

mentation of the story with the respective media and with which writers, photographers, and camera persons acquire the digital media content. Media designers edit and process the content for the targeted presentation. Finally, multimedia authors (often represented by computer scientists) compose and assemble the preprocessed and prepared material into the final multimedia presentation. This composition and assembly task is typically supported by professional multimedia development programs, so-called authoring software or authoring tools. Such tools allow for the manual (possibly assisted or wizard-based) composition and assembly of media elements into an interactive multimedia presentation via a graphical user interface. The result of this manual authoring process is typically a multimedia presentation, which is targeted only at a specific user or user group.

Even though we described this as a sequence of steps, the authoring process typically includes cycles. In addition, the expertise for some of the different tasks in the process can also be hold by one single person. This thesis is focusing on the part of the multimedia content creation process in which the prepared material is actually composed and assembled into the final multimedia presentation.

So far, we introduced and defined the notion of multimedia, presented the concepts of multimedia content, multimedia document models, and multimedia presentation formats. In the following section, we introduce the notion of personalization and discuss the impact that personalization has on the authoring of multimedia content.

3.1.4 Personalization

The World Wide Web has become a universal and borderless platform for the exchange and trade of information, services, and products [Spi00]. Its dimension and impact to our society could not necessarily been expected or foreseen to this extend [Bol05]. Today, arbitrary information, services, and products can be obtained and bought online. However, the personal contact between the providers and their users or customers that existed, e. g., in the traditional buyer-seller-relationship or when asking friends for some information or recommendation, is lost.

The nature of human beings likes to be individually treated, considered, advised, and entertained. In places, where no personal contact is available, e. g., in the web or in mobile situations, systems have to replace the task of appropriately offering individual information and communication by other humans. As a consequence, today's multimedia applications need to provide personalized content that actually meets the individual user's needs and requirements. This means that the multimedia content must reflect the user's situation, interests, and preferences, as well as the heterogeneous network infrastructure and (mobile) end device settings. Consequently, personalization is considered as a shift from a one-size-fits-all to a very individual and personal, i. e., one-to-one treatment [PR93] of customers and end users by the application [Bol03a].

The concept of personalization means different things to different people [Rie00a]. Consequently, a multitude of definitions can be found in literature, considering the notion of personalization from different point of views such as marketing, e-commerce, and computer science. The term personalization is often named together with the term of customization. However, the distinction between the both is not always clear [ELVKB01], are often intermixed, and sometimes considered equal

or interchangeable. In this work, we clearly distinguish between the notion of personalization and customization as done by [All99, Nie98, Rep03]. As we will see in the following discussion, this distinction is not a mere academic one but provides for defining two different application families, the customized applications and the personalized applications. This is due to the different requirements customized and personalized applications have to realize their functionality.

Customization is considered as an activity that is conducted and under direct control by the user [Nie98, All99]. This means that a customized application is actively adapted by their users to their individual needs and requirements. However, the customized application does not provide to adapt itself to the users. For example, users can customize their cell phone's user interface by selecting individual content such as ring tones, screensavers, and wallpapers. As customized applications are not able to adapt itself to the needs and preferences of the users, there is no need for them to provide a user model, i. e., to gather and maintain information or assumptions about the users' needs and preferences. All customization activities are carried out by the users themselves.

In contrast, personalization is considered as a process driven by the application [Nie98, All99]. Here, the content provided to the individual users is adapted by the personalized application itself. Thereby, the personalized applications take, e. g., the users' interests, preferences, and background knowledge into account [FK02, All99]. For example, a personalized web application tries to provide web pages to the user based on information and assumptions of the user's information needs [Nie98]. As a consequence, a personalized application must contain a user model, i. e., it must gather and manage at least some information or assumptions about the user's needs and requirements. With it, the personalized application is able to adapt itself to the needs and requirements of the user, i. e., to individualize [MAB00, Enc05, Mer06b] or tailorize [MAB00, Koc02] its content to the user model.

We support our decision to clearly distinguish between customized applications and personalized applications by the distinction traditionally made between adaptable systems and adaptive systems. While an adaptable system allows the user to change certain system parameters and adapt its behavior accordingly, an adaptive system can automatically change its behavior according to the information and assumption about their users [BRHO99, FKS97b]. Consequently, a personalized application is sometimes also called a user-adaptive application.

However, the border between customization and personalization is not always clear to draw as it may require an in depth insight of the application. For example, an automated flight-planning system introduced in [Hod02] requires the user to enter a destination and preferred dates, times, and airline. The flight-planning system then displays a list of flights matching the desired itinerary-including fare, seat availability, type of airplane, meals, and the like. In this case, it is ambiguous whether the flight-planning system employs a user model or not. In addition, applications can also provide for both customization and personalization, e. g., the web portal My Yahoo! [Yah06, MPR00]. Here, the web portal allows the users to select different portlets such as for news, weather, money, health, and entertainment to customize their portal content. However, some of these portlets also personalize their content to the information the users have provided the My Yahoo! portal, e. g., the local weather or some settings in presenting news.

On basis of the discussion about the notions of personalization and customization, we can now provide our definition of personalization. Within this work, personalization is defined as the offer and opportunity for a special treatment in form of information, services, and products, provided by an application according to the interests, background, role, facts, requirements, needs, and any other information and assumptions about an individual. The personalization of these information, services, and products is conducted pro-actively by the personalized application and typically carried out to the user in an iterative process.

From the nature of personalization, three necessary prerequisites can be derived for personalized applications. These are briefly described in the following:

Identifying the User First, the personalized application must be able to identify the user. Without knowing the identity of the user, the application cannot personalize its content and services to the needs and requirements of the individual user. For identifying the users, web-based personalized applications today typically request the users to register and then to log in by providing a user name and password. This enables to identifying the users and at the same time to protecting their private data. Here, the identification of the users is independent of where and from which device they access the application. However, identifying the users does not automatically mean that they need to register and log in by name and password. It is also possible to identify the users anonymously, e. g., by using web browser cookies. Here, however, a user moving from one device to another cannot be recognized anymore.

Implicit and Explicit Collection of Personalization Information Second, to be able to personalize its content, services, and products, a personalized application needs to have at least some information and assumptions about the user's needs and requirements. This information and assumptions can be acquired either by *explicitly* collecting or *implicitly* generating them [LNK04, BM02, CFMP99]. Here, the terms explicit personalization and implicit personalization are often used. With explicit personalization, the users are typically requested to fill out questionnaires when registering for a personalized service [CFMP99]. Consequently, the user gives information directly to the personalized application [BM02]. This is the case for example, when you register for the online bookstore Amazon [Ama06].

With implicit personalization merely the interaction of the user with the personalized application is observed [BM02]. Here, an interaction history of the user with the application is tracked, e. g., the mouse-clicks within a personalized web-based application [CFMP99]. Consequently, to obtain information and assumptions about the user's needs, implicit personalization requires the use of advanced techniques for processing and analyzing the gathered interaction data [LNK04]. Such a processing of user data is often referred as click-stream analysis. It allows personalized applications to extract and to derive behavior patterns of the individual user. An example for implicit personalization is [LL02].

Both approaches have their advantages and disadvantages. For example, with explicit personalization by filling out web forms it is more visible to the user what and which kind of information is collected. However, such questionnaires tend to annoy the users very fast. The strength of implicit personalization by, e. g., an-

alyzing web logs is that it is non-intrusive. However, it is difficult and requires sophisticated techniques to obtain reliable information about the user's needs (cf. [McC00]). Consequently, we often find a combined approach where initially some information about the user are collected explicitly by filling out a form and then the information and assumption about the user's needs are constantly updated by implicit personalization techniques. This takes place, e. g., with the online bookstore Amazon [Ama06] mentioned above.

Meta Data of the Personalized Information, Services, and Products Third, in order to provide for personalized information, services, or products, the personalized applications not only need to have at least some information and assumptions about the user's needs. In addition, to provide for personalization the objects, i. e., the information, services, and products, must carry a sufficient amount of meta data. This meta data is exploited by the personalized application in order to provide for personalized information, services, or product recommendations to the users according to their needs and requirements. For example, meta data a online bookstore requires are information about at least the content and subject of the books it sells in order to meet the user's interests for a specific topic. Today, we are pleased to observe that there exist personalized systems and applications for a wide range of application domains. For example in the domain of online-recruitment [SBR02], finance and banking [DFG+04, WW00], medicine [BC02], (television) news [FKNS02, MLLR99], (mobile) tourism [PKK01, CMD02], e-learning and education [DB02, BSZM02, WB01, Kle06], entertainment [HLZ04], web portals [Yah06], electronic catalogs [AGPS02, LLHC01], and e-commerce [Ama06].

Levels of Personalization The levels of personalization applications provide today are wide-ranging [KNV00]. With name recognition a simple personalized greeting in web sites and e-mails like "Hello Mr. Jones, ..." is provided, e. g., by the business platform openBC [Ope06a]. Check-box personalization (also called user-provided information) uses explicit information about the users' interests and preferences, gathered from questionnaires and registration forms. With segmentation and rules-based processing users are divided from a large population into smaller user groups [Pey03] based on information about demographic, geographic, psychographic and other data. (Behavioral) preference-based personalization employs implicit personalization techniques to derive some specific interests users could have in order to provide them with well-directed offers. Having discussed and defined the notion of personalization in this section and introduced the notion of multimedia content and content authoring in the previous sections, we introduce the notion of personalized multimedia content in the following section.

3.1.5 Personalization of Multimedia Content

Based on the definition of personalization and multimedia content, we consider personalized multimedia content as multimedia content targeted at a specific user or user group. It reflects the individual user's or user group's needs, background, interests, and knowledge, as well as the heterogeneous infrastructure of the (mobile) end device to which the content is delivered and on which it is presented. Consequently,

the authoring of personalized multimedia content is considered as a process of selecting and composing media elements into personalized multimedia content, i. e., into a coherent, continuous multimedia presentation that best reflects the needs, requirements, and system environment of the individual user.

The authoring process of multimedia content described in Section 3.1.3 represents a manual authoring of such content, often involved with high effort and cost. Typically, the result is a multimedia presentation targeted at a certain user group in a specific technical context. However, this one-size-fits-all fashion of the multimedia content created does not necessarily satisfy the different users' needs. Different users may have different preferences concerning the content and also may access the content in networks on different (mobile) end devices. Consequently, the authoring of personalized multimedia content raises new requirements. For a wider applicability, the authored content needs to "carry" some alternatives or variants that can be exploited to adapt the multimedia presentation to the specific preferences of the users and their technical settings. For example, Figure 1.3 in the introduction of this work shows the variation possibilities that are already involved with a simple personalized city guide application. In the following section, today's support for the authoring of personalized multimedia content is presented.

3.2 Analysis of Existing Authoring Support for Personalized Multimedia Content

In the following, we present the state-of-the-art in the field of personalized multimedia content authoring and analyze existing systems and projects for multimedia content adaptation and personalization. We present today's support for personalized multimedia authoring from different views and aspects. We begin with adaptive multimedia document models, followed by the manual authoring of (personalized) multimedia content by means of appropriate authoring tools. Along the way from manual authoring of multimedia content to dynamic authoring of personalized multimedia content, we consider distinct systems, projects, families of systems, and research directions. Among this are text-centric systems, projects that focus on single media elements, provide support for adaptive interaction, allow for authoring of mobile multimedia content, and personalized multimedia content. Following this fairly broad overview and analysis of today's support for personalized multimedia content, the considered systems, projects, families of systems, and research directions will be categorized in different approaches for multimedia personalization in Section 3.3.

3.2.1 Adaptive Multimedia Document Models

Early work in the field of creating advanced hypermedia documents and multimedia documents can be found, e. g., with the Amsterdam Hypermedia Model (AHM) [Har98, HBR94] and the authoring system CMIFed [Bul95, RJMB93, HRJM94], working with events and timing-constrains. Constraints are also employed by the adaptive document model provided with the multimedia authoring environment Madeus [JLR+98, JLSI97] (see also Section 3.2.2.2).

The ZYX [BK01, Bol01] multimedia document model provides—besides a temporal, spatial, and interaction model for multimedia content composition—an extensive support for reuse (of parts) of multimedia presentations and adaptability of the multimedia content. Here, an extensive analysis in the field of multimedia document models and formats has been conducted to bring together the benefits of the different models and formats. The ZYX model is used, e. g., for the content authored with the domain-specific authoring wizard Cardio-OP [KGF99] presented in Section 3.2.2.1.

In the field of standardized document models, the declarative description of multimedia documents with SMIL allows for the specification of adaptive multimedia presentations by defining presentation alternatives using the switch element. Some SMIL tools such as the GR*i*NS editor presented in Section 3.2.2.2 provide support for the switch element to define presentation alternatives; a comfortable interface for editing the different alternatives for many different contexts, however, is not provided. To close this gap, there has been work done on an approach, where a multimedia document is authored for one general context and is then "automatically" enriched by the different presentation alternatives needed for the expected user contexts in which the document will be viewed [BKW99a]. However, this approach is reasonable only for a limited number of presentation alternatives and limited presentation complexity in general.

3.2.2 Multimedia Authoring Tools

Multimedia authoring tools allow for the manual (possibly assisted or wizard-based) composition and assembly of media elements into an interactive multimedia presentation via a graphical user interface. For creating the multimedia content, the authoring tools follow different design philosophies and metaphors, respectively. These metaphors can be roughly categorized into script-based, card/page-based, icon-based, timeline-based, and object-based authoring [RB96]. All these different metaphors have the same goal, to support authors in manually creating their content. Looking at the field of multimedia authoring tools, we find a range from highly specialized domain expert authoring tools to generalized multi-purpose authoring tools.

3.2.2.1 Highly Specialized Domain Expert Authoring Tools

Domain expert tools hide as much as possible the technical details of content authoring from the authors. They let them concentrate on the actual creation of the multimedia content. The tools we find here are typically very specialized and targeted at a very specific domain. They are often organized in a wizard-like fashion that guide the expert in the field through the authoring process.

An example of a highly specialized domain expert authoring tool that supports the creation of personalized multimedia content is the page-based Cardio-OP Authoring Wizard [KGF99]. The wizard provides support for personalized content authoring in the field of cardiac surgery [KGF99, GR98, BKHW01]. It enables medical experts to compose an interactive multimedia book on operative techniques for three target groups, which are medical doctors, nurses, and students.

The Authoring Wizard guides the domain experts through the particular authoring steps by a digital storyboard for a multimedia book on cardiac surgery. It offers dialogs specifically tailored to the needs of each step and hides as much technical details as possible. Coupled tightly with an underlying media server, the authoring wizard allows use of every precious piece of media data available at the media server in all of the instructional applications at different educational levels. This promotes reuse of expensively produced content in a variety of different contexts. Personalization of the e-learning content is required here, since the three target groups have different views and knowledge about the domain of cardiac surgery. Therefore, the target groups require different information from such a multimedia book, presented on an adequate level of difficulty for each group.

3.2.2.2 Generalized Multi-purpose Authoring Tools

On the other end of the spectrum, we find highly generalized multimedia authoring tools such as Adobe's Authorware [Ado05b], Director [Ado05a], Flash Professional [Ado06b], and Toolbook [Sum06]. These domain-independent tools let the authors create very sophisticated multimedia presentations in a proprietary format. However, the generalized multi-purpose authoring tools today typically require high expertise in using them. Typically, they do not provide for personalization support. Everything "personalizable" needs to be programmed or scripted within the tool's programming language. Consequently, the multimedia authors need programming skills and along with this some experience in software development and software engineering.

However, some first steps how personalized multimedia content could be authored by generalized multi-purpose authoring tools, shows the icon-based authoring tool Authorware [Ado05b]. This tool allows an icon-based respectively flow chart oriented creation of multimedia content and supports the automatic control of the presentation's flow with its Decision Icon. The Decision Icon calculates the current value of a variable or expression that is attached to it and determines by this means which path of the decision structure is to be followed [Ado05b]. Thus, the Decision Icon can be used in principle to "personalize" the flow chart of a multimedia application developed with Authorware. However, an enhanced support for developing personalized applications is not provided.

In the field of adaptive multimedia models there exist authoring tools such as the GR*i*NS editor [BHJ+98, Ora06b] from Oratrix for SMIL documents (see also editing of SMIL's switch elements in Section 3.2.1). Another tool that provides for the authoring of personalized multimedia presentations is the Madeus authoring environment [JLR+98, JLSI97]. Here, constraints are exploited to compose and assemble adaptable multimedia presentations. However, these constraints provide for a personalization support only within a limited range and do not support exchange of presentation fragments as it is supported, e. g., by SMIL's switch element (cf. Section 3.2.1). In addition, the generalized authoring tools are still tedious to handle and not practical for the domain experts, and following Bulterman in [BH05]: "Unfortunately, we have not seen the hoped-for uptake of authoring systems for SMIL or for any other format."

In order to provide multimedia authors a comprehensive support for personalized multimedia content, the multi-purpose authoring tools need to offer an editor

that explicitly allows to define (abstract) user profiles. These user profiles need then to be matched with, e. g., Authorware's Decision Icon, in order to select those paths respectively alternatives of the flow chart that best fits the users.

3.2.3 Adaptive Hypermedia and Intelligent Tutoring Systems

On the step towards the creation of personalized *multimedia* content, we find interesting work in the area of adaptive hypermedia systems (AHS) which is going on for quite some years now [Bru96, Bru94, WKDB01, DBBH99, CP01, DBAHW00, DBBC02, DCRA+98, DCRBM99]. The adaptive hypermedia system AHA! [DBHW99, DBASS02, DBAB+03] is a prominent example, which also addresses the authoring aspect [SDB03], e. g., in adaptive educational hypermedia applications [SCDB04].

However, though these and further approaches integrate media elements in their adaptive hypermedia presentations, synchronized multimedia presentations are not in their focus. The main personalization techniques pursued are adaptive navigation support and adaptive presentation. With adaptive navigation [DBBH99] links are, e. g., enabled, disabled, annotated, sorted, and removed, according to the profile information about the user (also called link-adaptation [DBHW99]). The purpose of adaptive presentation [DBBH99] is to, e. g., show, hide, reorder, and highlight or dim specific fragments of the presented hypermedia content according to the user profile information (also called content-adaptation [DBHW99]).

Closely related to adaptive hypermedia systems are the so-called intelligent tutoring systems (ITS) [Sch97]. ITS provide personalized content according to the learners or students knowledge. The aim of ITS is that the learners gain new knowledge and skills in a specific domain by independently solving problems in that domain. An ITS provides a model of the student, a model of the knowledge domain, and modeled educational strategies [Sch97]. This means, it comprises explicit assumptions and information about the knowledge and level of knowledge of the user in the considered domain (student model or diagnosis model), an expert's knowledge in the domain (domain model or expert model), and a didactic concept of how to convey and present the learning materials to the learners (tutor model or educational model). Such models are also defined with adaptive hypermedia systems, e. g., [DBAB+03, WDB02, WKDB01], although they are named differed there. Consequently, AHS are sometimes also regarded as an integration of ITS and hypermedia systems, e. g., in [Con00].

3.2.4 HotStreams

We also find a very interesting approach with the interactive video application framework proposed in [HVL01] and its concrete implementation, the HotStreams system. Within the HotStreams system developed by Siemens personalization takes place in regard of individually selecting and composing video clips into a personalized video stream. This video stream can be accumulated by personally selected advertisement clips. In addition, personalized interactivity is supported by providing individual hotspots and hyperlinks to the user. For actually merging the individually selected video clips, advertisements, and navigational interaction possibilities into the personalized video stream, SMIL templates are filled and send to the user.

3.2.5 COMET and MAGIC

A very early approach towards the dynamic authoring of adapted multimedia content is the Coordinated Multimedia Explanation Testbed (COMET), which bases on an expert system and different knowledge databases and uses constraints and plans to actually generate the multimedia presentations [FM93, MRT93, EFMS91]. Another approach that goes into the same direction is MAGIC (Multimedia Abstract Generation for Intensive Care) [DFM+96]. MAGIC uses expert systems with static knowledge bases and a content planer.

3.2.6 WIP and PPP

A very interesting approach to automate the personalization of multimedia content has been developed at the DFKI in Germany by the two knowledge-based systems WIP (Knowledge-based Presentation of Information) and PPP (Personalized Plan-based Presenter). WIP is a knowledge-based presentation system that automatically generates instructions for the maintenance of technical devices by plan generation and constraint solving. PPP enhances this system by providing a life-like character to present the multimedia content and by considering the temporal order in which a user processes a presentation [AMR96b, AMR96a, AR96, AR95, AFG+93].

3.2.7 Cuypers Multimedia Transformation Engine

Also a very interesting research approach towards the dynamic generation of multimedia presentations is the Cuypers Multimedia Transformation Engine [LH04, OHGR03, GOH01, OCG+00, Geu02] developed at the CWI. The Cuypers system is a generic application employing constraints for the description of the intended multimedia programming and logic programming for the generation of the multimedia content [GOH01]. This multimedia content is represented by and assembled in Hypermedia Formatting Objects (HFO) [OHGR03]. To actually present these HFOs, they are transformed into a SMIL presentation by applying an appropriate XSL style sheet [ABC+01a].

3.2.8 Opéra and WAM

The multimedia document group at INRIA in France developed within the Opéra project a generic architecture for the automated construction of multimedia presentations based on transformation sheets and constraints [Vil01, BJK01]. This work is continued within the succeeding Web, Accessibility, and Multimedia project (WAM) with the focus on a negotiation and adaptation architecture for multimedia services for mobile devices [LL03b, LL03a, LL04]. Further authoring support for mobile multimedia content is presented in the following section.

3.2.9 Mobile Multimedia Content

In regard of personalized mobile multimedia content, an interesting approach can be found with the adaptive multimedia middleware architecture developed in the

Princess project, which allows for the adaptation of media elements and user interfaces to different mobile devices [MLK+01]. Within this middleware architecture, the content is represented in an extended version of XHTML [KPMM01]. The content is carried out by the middleware into different formats such as HTML [RLHJ99] and WML [Ope01] employing different technologies. For example, XSLT [ABC+01a, W3C99] is used to transform the XHTML-based content into WML [KPMM01].

An approach that deals with the adaptation of streamed media content to mobile devices is the Media Stream Transcoding Project [Com06]. Here, high-quality media streams are transcoded to the capabilities of different mobile end devices. Another approach dealing with the transcoding of single media streams is the koMMa framework [JLTH06, LJH04]. The koMMa framework acts as a proxy for carrying out the personalized audiovisual media streams from the content providers to the (mobile) end users. It employs MPEG-7 [ISO99, ISO01f, ISO01b, ISO01c, ISO01d, ISO01e] and MPEG-21 [BWH+03] for the media content description and adaptation task. Also within the RETAVIC project [Fri06], a media transformation framework for audiovisual media content has been developed. Distinctive feature of this approach is the internal video format called LLV1 [SMMW05]. The LLV1 format abstracts from today's video formats and aims at a lossless storage and processing of arbitrary audiovisual media content. It is independent of the format in which the audiovisual content is originally recorded as well as independent of the format in which it is deployed to the (mobile) clients.

In the more particular area of location-based services we find some tourist guide applications that deal with the adaptation of mobile content. The mobile city guide GUIDE [CMD02, DCMF99] for the city of Lancaster, UK, was one of the first systems to integrate personalized information to the user. Here, hypertext-based information is presented to the user, which is authored by employing templates augmented with some selection instructions in form of decision-rules to personalize the content [DCMF99]. Another mobile tourist guide application is LoL@ [UPNM03, PUM02, PKK01] for the city of Vienna, Austria. Here, XML-based templates are used to author the personalized content and XSLT is applied for transforming the content to HTML or WML.

3.2.10 Multimedia Calculi and Algebras

So far, only few work has been conducted in the area of multimedia composition by calculi and algebras (cf. [Sub98]). One example is the multimedia presentation algebra (MPA) by Adali et al. [ASS00, ASS99]. This algebra extends the relational model of data and allows for dynamically creating new presentations from (parts of) existing presentations. With the MPA, a page-oriented view on multimedia content is given. A multimedia presentation is considered as an interactive presentation that consists of a tree, which is stored in a database. Each node of this tree represents a non-interactive presentation, e. g., a sequence of slides, a video element, or a HTML page. The branches of the tree reflect different possible playback variants of a set of presentations. A transition from a parent node to a child node in this tree corresponds to an interaction.

The proposed MPA allows for specifying a query on the database based on the contents of individual nodes as well as querying based on the presentation's tree

structure. For it, the MPA provides extensions and generalizations of the select and project operations in the relational algebra. However, it also allows to author new presentations based on the nodes and tree structure stored in the database. For it, the MPA defines operations such as merge, join, path-union, path-intersection, and path-difference. These extend the algebraic join operation to tree structures and allow to author new presentations by combining existing presentations and parts of presentations.

Another research approach comprises a multimedia calculus and algebra allowing for querying on tree-based multimedia content stored in multimedia databases [LSB+99, LSB+00, LO96]. Here, the new multimedia presentations are created on the basis of a given query and a set of inclusion and exclusion constraints stored in the database.

3.2.11 Standard Reference Model for Intelligent Multimedia Presentation Systems

Finally, we find with the Standard Reference Model for Intelligent Multimedia Presentation Systems (SRM for IMMPS) [SRM97, BFF+97, BFR+96, FR96] a generalized architecture for the domain of so-called intelligent multimedia presentation systems (IMMPS). The aim of IMMPS is to automate the authoring of multimedia presentations in order to enable on-the-fly personalization of presentations according to the individual needs of the user [SRM97]. Hereby, IMMPS exploit techniques originating from the research area of artificial intelligence (AI) [RN03] such as knowledge bases, planning, user modeling, and automated generation of media elements such as text, graphics, animation, and sounds [SRM97]. The goal of the SRM for IMMPS is to provide a common framework for the analysis, comparison, and benchmarking of IMMPS. The initial proposal of the SRM for IMMPS has been discussed during the ECAI'96 Workshop [FR96] by researchers within the AI research community. A revised version of the proposal [BFR+96] was presented to researchers from the multimedia community at the Multimedia Modelling Conference in the same year. An example of a system employing the SRM for IMMPS is the Berlage environment providing for a dynamic authoring of adaptive hypermedia content [RHOB98, ROHB98]. For the internal representation of the provided hypermedia content, the Berlage environment employs the Hypermedia/Time-based Structuring Language (HyTime) [ISO97]. It uses Document Type Definition (DTD) [W3C04] and Document Style Semantics and Specification Language (DSSSL) style sheets [ISO96] for encoding the hypermedia content in the final presentation format SMIL.

The SRM for IMMPS describes the structure of personalized multimedia applications on a high level. It describes different components and the communication between the components. However, the SRM for IMMPS does not provide a concrete software engineering support for actually designing and implementing IMMPS. This means that the SRM for IMMPS cannot directly be applied for an industry like development of personalized multimedia applications. As a consequence, the application developers need to design large parts of the application by themselves. Although we find with the Berlage environment an implementation of the SRM for IMMPS that provides for a readily implementation of adaptive hyperme-

dia applications [BRHO99], these applications are fixed to the concrete technological setting used for the Berlage environment described above.

3.3 Systematic Categorization of Multimedia Personalization Approaches

The related work for the authoring of personalized multimedia applications shows that there are different technologies employed to provide personalized content to the users. Analyzing the different solutions and projects for multimedia personalization, one can derive some general solution approaches pursued for the challenging task of multimedia personalization. Defining such general approaches for multimedia personalization and classifying the existing systems and projects into these approaches is a very difficult and challenging task. Thus, it is not very surprising that there is only little work done in classifying and comparing the different systems and projects. To the best of one's knowledge, the only known sources in literature is the work by Jourdan et al. [JB01, JLR99]. They provide a classification of existing systems and projects into different groups and valuate these groups. In this work, we modify and extend the classification proposed in [JB01, JLR99] by suggesting the following six categories for classifying today's multimedia personalization approaches:

- personalization by programming,
- personalization by templates and selection instructions,
- personalization by adaptive document models,
- personalization by transformations,
- personalization by constraints, rules, plans, and knowledge bases, as well as
- personalization by calculi and algebras.

When considering the existing systems and projects, it is not always easy to decide to which of the proposed categories a specific solution should be associated. In addition, some of the existing systems and projects explicitly combine or *synthesize* different approaches and thus need to be associated to more than one category. This means that they employ more than one approach to actually realize their personalized multimedia functionality. Examples for such hybrid systems are [JB01] and [KYME03]. Consequently, the presented categories must be considered as coarse grained grouping and classification of the existing support of multimedia personalization by the different systems and projects rather than an exact sharing out among the proposed categories.

The single approaches for multimedia personalization are described in the following Sections 3.3.1 to 3.3.6. For each personalization approach, we first introduce the characteristics of this approach. Then, we refer to representative systems and projects for the considered approach. Finally, a valuation of the approach is given, i. e., the advantages and disadvantages of the approach are discussed. An overview of the valuation of the personalization approaches concludes this section.

3.3.1 Personalization by Programming

For the first category, the personalization by programming approach, regular programming languages are applied to develop the (multimedia) personalization functionality. Obviously, programming languages such as C++ and Java can be employed to develop a system that generates personalized multimedia content [JB01]. However, also logic programming, e. g., with Prolog, can be used to realize the personalization functionality.

Examples Programming is typically employed with the generalized multi-purpose authoring tools presented in Section 3.2.2.2 to provide for personalized multimedia content. Here, languages that are applied are Director's Lingo [Ado05a] or Flash's ActionScript [Ado06b]. However, there exist with, e. g., the Madeus authoring environment and CMIFed, also generalized authoring tools that employ constraints for carrying out adaptable multimedia content. Besides the generalized authoring tools, probably also most specialized authoring tools today are programmed, such as the presented Cardio-OP Authoring Wizard presented in Section 3.2.2.1.

Programming is also employed for distinct personalization tasks such as layouting of visual media elements. For example, we find a straightforward layout adaptation algorithm with [BV03] and a layout approach that employs genetic programming in [GL03].

Valuation With mere programming, every personalization functionality is feasible that can be implemented with a programming language. However, a well known disadvantage of this approach is the lack of independence between the programming code and the piece of information [JB01]. This makes it difficult to reuse some parts of an application for another one. In addition, as providers today typically develop their own specific solution, the provided personalization functionality is tailored and designed for their particular application domain. As a consequence, high effort is necessary when extending or adapting the solution to a different domain. Finally, as there is none or hardly reuse between the individually developed systems for multimedia personalization, it is very difficult if not unfeasible to integrate them.

3.3.2 Personalization by Templates and Selection Instructions

With the second category, personalization takes places with templates and selection instructions. A template can be considered as the static part of a multimedia presentation which is possibly designed in a concrete presentation language such as HTML or SMIL and that is enriched with some selection instructions [JB01]. These selection instructions are executed when the user requests the template. Executing selection instructions typically means that information is extracted from external data sources [JB01] according to the user's profile information. The dynamically extracted information is then merged on-the-fly with the template. Merging the static part of the multimedia presentation with individually selected dynamic content on-demand allows the templates approach to provide for a personalized composition and delivery of information to the users.

Examples The mobile tourist application GUIDE presented in Section 3.2.9 exploits templates augmented with some selection instructions in form of composition rules to provide its personalized hypermedia information. The mobile city guide LoL@ also presented in Section 3.2.9 uses XML-based templates and employs XSLT to author the personalized mobile content. Another approach working with templates is the HotStreams system presented in Section 3.2.4. Here, the personalized audiovisual content is dynamically generated by enriching SMIL templates with video clips according to the profile information about the user. An approach for solving layout problems with adaptable templates employing constraints is presented in [JLS+04].

Valuation The template-based approach for multimedia personalization suits well for applications in which content selection can be split into several database requests [JB01]. However, such static templates are limited in their expressiveness. In addition, it is possibly very difficult to handle global content selection and assembly criteria when considering multiple database requests in order to choose those media elements that, e. g., best reflect the user's profile information or do not cross a maximum time limit of the presentation duration [JB01]. This could be implemented through successive database requests. However, the time performance of such a personalized application may become very slow.

3.3.3 Personalization by Adaptive Documents Models

Personalization by adaptive multimedia document models provides for specifying different presentation alternatives or variants of the multimedia content within the multimedia document. The presentation alternatives are statically defined within the adaptive multimedia document. During presentation time, the document's alternative or variant is determined that best matches the user's profile information. Finally, the selected presentation alternative is presented on the end device. Consequently, with the personalization approach by adaptive document models, the multimedia player on the (mobile) end device decides within the range of the available presentation alternatives which variant of the multimedia document is presented.

Examples Examples for adaptive document models are, e. g., the Amsterdam Hypermedia Model, SMIL, and ZYX, presented in Section 3.2.1.

Valuation Adaptive multimedia document models provide for an extensive support for the adaptation and reuse of multimedia presentations and parts of it. Another advantage of this approach is that there are W3C standards, such as SMIL. However, adaptive document models are not practicable when it comes to a comprehensive support for personalization, as all different presentation alternatives need to be specified in advance within the same document (cf. the variation possibilities involved with our tourist guide application sketched in Section 1.1).

3.3.4 Personalization by Transformations

With the personalization by transformations approach two kinds of transformations are involved [LL03b], the structural transformation and the media transformation. With structural transformation is meant to change the structure of the multimedia content. Here, the multimedia content is typically represented in a data structure or document based on XML [W3C04]. A structural transformation can be, e. g., to transform a XML-document into a (standardized) multimedia presentation format such as SMIL or SVG. Structural transformations also include changing the layout and arrangement of the media elements to different presentation styles (cf. personalization by style sheets in [JB01]), e. g., changing the spatial layout of the visual media elements. However, with structural adaptation, also a XML-document can be adapted to be presented on a Desktop PC and a mobile device, e. g., by dividing the content into different smaller screens or pages in the mobile situation. Consequently, structural transformations typically implicate an adaptation of the temporal and/or spatial layout of the multimedia presentation as well as possibly changing the interaction design of the presentation with the user.

Considering media transformation, this kind of transformation is concerned with changing the media type, e. g., switching from an image element to a text element describing the same content. It also includes adapting the media format, e. g., transcoding an image element from PNG to JPG, and conducting other binary operations on the media element such as resizing a video element or changing the color-depth of an image element.

Examples As XML is typically employed to represent the multimedia content for the personalization by transformation approach, very often XSL is used to actually transform the content. XSL consists of two parts, XSLT and XSL-FO. XSLT is used to change the structure of the multimedia document, while XSL-FO targets at providing for different presentation styles such as it is provided by Cascading Style Sheets (CSS) [W3C06a]. For media transcoding and other binary operations on the media elements external libraries and systems are employed.

Prominent examples for the personalization by transformation approach are the projects Opéra and WAM, presented in Section 3.2.8. Here, XSLT is employed for structural transformations. External media transcoders are used for media transformations. Transformations are also used by the Cuypers multimedia presentation engine presented in Section 3.2.7 to transform the multimedia content represented in their HFOs into the final multimedia presentation format. Another system that employs transformation is the adaptive multimedia middleware architecture presented in Section 3.2.9. XML and XSLT are employed to generated SMIL documents within the Course Authoring and Management System (CAMS) [CKAH02] in the domain of e-learning. The automatic generation of hypermedia content with the KIWIS system [VOGME02] employs XSL. Another approach employing XSL transformation for multimedia presentation generation is the Hera design method [FHBP03].

Approaches that focus on media transformations are typically found in the area of mobile multimedia presentation generation, as to be found with the Media Stream Transcoding Project, the koMMa framework, and the RETAVIC project's LLV1 approach presented in Section 3.2.9. Other approaches for transcoding and manip-

ulating media streams are the research solutions Network-Integrated Multimedia Middleware [Com05] and Presentation Processing Engine [PLV97, PVL96] as well as the industrial solutions DirectShow [Mic06a] by Microsoft, Helix DNA [Rea05] from RealNetworks, and QuickTime [App05] by Apple.

Valuation One advantage of the personalization by transformation approach is that the adaptation of the multimedia content can be described in the W3C standard XSL, employing XSLT and XSL-FO. With [Uni01] a touring machine exists for XSLT. Thus following the Church–Turing thesis [EB90, Cou90, Sal89], XSLT is computational complete, i. e., arbitrary transformations can be conducted with XSLT that can be described by an algorithm.

For example, with XSLT it is possible to describe transformations between different presentation formats, e. g., HTML and WML [Oes04]. However, due to recursive structures involved with it, such transformation descriptions are very complex and difficult to handle. Having a multitude of sequentially executed (simpler) transformations is also not easy to handle, as there is no encapsulation mechanism available in XSLT that provides for hiding some transformation details at higher levels.

3.3.5 Personalization by Constraints, Rules, Plans, and Knowledge Bases

With this approach, personalization takes place by using constraints, rules, plans, and knowledge bases. Here, creating the personalized multimedia content is typically considered as optimization problem [RN03] where constraint solving needs to be applied to find the optimal selection and arrangement of content. The creation of the personalized multimedia content is explicitly described by using rules, constraints, and the like, which are, e. g., stored in different knowledge bases. The personalized multimedia presentation is generated on basis of such a declarative description and by taking the profile information about the user into account. Such a presentation generation can also be regarded as planning problem [RN03]. Here, the user's request is decomposed in some subgoals to reach. The results of these subgoals are then accumulatively assembled together to the final multimedia presentation (cf. [JB01]).

Examples A prominent example of a system that employs constraints and rules for the personalized multimedia content generation is the Cuypers Multimedia Transformation Engine presented in Section 3.2.7. Although, it also employs transformation sheets, the main means for generating the personalized multimedia content are constraints and rules. With COMET and MAGIC, we find two knowledge-based systems for personalization, presented in Section 3.2.5. Further systems are WIP and PPP described in Section 3.2.6, as well as the research area of adaptive hypermedia and intelligent tutoring systems considered in Section 3.2.3. With the Standard Reference Model for Intelligent Multimedia Presentation Systems, we find a very generic approach for the so-called intelligent multimedia presentation systems. Here, multiple knowledge bases are exploited for the personalization task as well as constraints for layouting. The authoring environments Madeus and CMIFed described in Section 3.2.2.2 employ constraints for determining adaptable multime-

dia presentations. In [CFMP99] the well known event-condition-action rules are used for a one-to-one web personalization. Here, the rules can have different priority. Constraints-based personalization of multimedia content is also provided by the Nsync toolkit [BKCD98, BK97]. It provides for specifying the synchronization and interaction properties of the multimedia presentation. In the area of personalized media summaries there exists, e. g., a rules-based approach for generating personalized music sports videos [WXC+05] and IBM's Video Semantic Summarization System providing personalized summaries of videos taking time-constraints and display size constraints into account [IBM04a]. With the multinational project SmartKom [Wah02, WRB01, DFK06], we find a very interesting and prominent approach for providing personalized multimedia content via multimodal input and output devices. One result of this project is the intelligent tourist guide Deep Map [MZ00], providing personal guided walks for tourists through the city of Heidelberg in Germany by employing knowledge bases and planning systems.

Constraints and knowledge bases are also often used to solve layouting tasks in personalized applications. Very first work in this area is LayLab [Gra95], a constraint-based layout manager for multimedia presentations. This layout manager is employed, e. g., for the two systems WIP and PPP presented in Section 3.2.6. Another work conducted in the area of automatic layouting is the generic model for Intelligent Multimedia Layout Manager (IMMLM) [Gra97] developed in the context of the SRM for IMMPS. Further work is the support for a knowledge-based adaptive layout of dynamic web pages by Kröner [Krö01] and the Intelligent Multimedia Application GENerator (IMAGEN) project [KS04, RKB02]. The latter provides a tool set for the creation of personalized presentation services in the Internet.

Valuation Systems applying personalization by constraints, rules, plans, or knowledge bases work on a declarative level describing the personalization functionality. However, due to their declarative description languages, only those multimedia personalization and generation problems can be solved that can be covered by such a declarative specification. Consequently, these systems and projects find their limits when it comes to more complex or application-specific multimedia personalization functionality and additional programming is required to solve that problem.

3.3.6 Personalization by Calculi and Algebras

With the last solution approach, calculi and algebras are applied to select media elements and merge them into a coherent multimedia presentation. This approach emerged from the database community with the aim to store, process, and author multimedia presentations within databases. Consequently, work based on calculi and algebras are applied on the database level and provide for specifying queries that are send to a database system. The database system executes the queries and determines the best match of the different media items and presentation alternatives stored in the database. The result is then send back to the querying application.

Examples Examples of calculi and algebras providing for the querying and automatic assembly of multimedia content such as the multimedia presentation algebra are presented in Section 3.2.10.

Valuation The main advantage of the personalization by algebras approach is that the requested multimedia content is specified as a query in a format language. However, typically high effort is necessary to learn the algebra and their operators. Consequently, it is very difficult to apply such a formal approach.

3.3.7 Summary of Multimedia Personalization Approaches

The classification of the existing systems and projects to the different categories of personalization approaches is not always easy and unambiguous. Nevertheless, we presented a categorization of today's support for the authoring of personalized multimedia content. This provides for a more systematic management and examination of the tasks and challenges involved with the creation of such content. After having introduced the different personalization approaches, representative systems and projects have been referred and presented, respectively. In addition, a valuation of each approach is conducted. Table 3.1 summarizes the valuation and provides a basis for some conclusions for the envisioned MM4U approach. These are presented in the following section.

3.4 Conclusions for the MM4U Approach

In the previous sections, we presented the notion of authoring (personalized) multimedia content, analyzed existing systems and projects in the field of multimedia content personalization, identified and valuated different approaches for multimedia personalization, and categorized the existing solutions into the identified approaches. From this analysis and categorization, we conclude that the envisioned MM4U approach needs to fulfill the following requirements.

- Provide support for a dynamic authoring of personalized multimedia content.
- Provide a generic software engineering support to develop personalized multimedia applications.

Need for a Dynamic Authoring of Personalized Multimedia Content The example of our personalized tourist guide presented in Section 1.1 shows the variation possibilities involved with the authoring of personalized multimedia content. As it is illustrated there, the number of variants explodes with an increasingly comprehensive personalization support. From our point of view, a manual authoring of personalized multimedia content is not feasible due to the huge variety of different variation possibilities involved [AMR96b, SB05c]. Hence, the creation of personalized multimedia content needs to be automated and carried out dynamically [SRM97]. Only by providing support for a dynamic authoring of personalized multimedia content, we will be able to best reflect the individual user's needs and requirements.

Personalization approach	Valuation
Programming	+ Arbitrary personalization functionality is feasible − Providers develop their own (mostly) complex solution − High effort necessary for extending or adapting the solution to a different domain
Templates and selection instructions	+ Suits well for applications in which content selection can be split into several database requests − Static templates are limited in their expressiveness − Difficult to handle global content selection and assembly criteria
Adaptive document models	+ Extensive support for adaptation and reuse of multimedia documents and parts of them + W3C document standards exist − Not practical since all presentation variants need to be specified in advance
Transformations	+ Description of content and transformation in W3C standard + XSLT is calculation complete − Difficult to handle multiple or complex transformations
Constraints, rules, plans, and knowledge bases	+ Declarative description of the personalization functionality − Power of expression limited to declaratively describable personalization problems − Additional programming is unavoidable for complex and application-specific personalization functionality
Algebras	+ Formal description of the multimedia composition − High effort necessary to learn the algebraic operators − Application of the formal operators is difficult

Table 3.1: Evaluation of the six categories of multimedia personalization approaches

Need for Pursuing a Software Engineering Approach Summarizing the existing systems and projects, we see that most apply to text-centered information only. Many of these systems for content personalization are targeted at a specific application domain in which they provide a very specific content personalization task. The existing systems and approaches typically use declarative languages, e. g., rules and constraints, or exploit style sheets, transformation sheets, templates, and the like, to express the dynamic, personalized multimedia content creation. Additionally, the systems and applications we find today usually rely on fixed data models for describing user profiles, structural presentation constraints, technical infrastructure, rhetorical structure, and the like. A change of the input data models as well as an adaptation of the presentation generator to more complex presentation generation tasks is difficult if not unfeasible. In addition, the border between the declarative descriptions of the content personalization constraints, rules, and the like and the additional programming employed is not clear and differs from solution to solution.

Consequently, we observe that the existing approaches for multimedia content personalization find their limits in regard of required effort, applicability and prac-

ticability, or power of expression. Whenever a complex and application-specific personalization generation task is required, additional programming becomes unavoidable anyway to solve the problem. However, if programming is unavoidable for complex or application-specific personalization tasks, the question raises for providing a unified software engineering support for creating personalized multimedia content.

Consequently, we pursue with the MM4U framework [SB05c, SB05d, SB04b] a software engineering approach that provides generic and domain-independent support for the development of personalized multimedia applications. With its components, the MM4U framework provides comprehensive support for the typical tasks of personalized multimedia authoring. The framework relieves application developers from the general tasks in the context of multimedia content personalization and lets them concentrate on the application-specific tasks. It does not re-invent multimedia content adaptation but is targeted at incorporating and embedding existing research approaches in the field and allows to be extended by domain-specific and application-specific solutions. We support our decision for a software engineering approach by the assumption that analyzing personalization from a software engineering point of view will allow for designing more modular personalized applications, which are easier to maintain and to extend [RSG01]. In addition, as we have shown in Section 3.3, each personalization approach has its individual advantages as well as disadvantages. By pursuing a software engineering approach that allows for integrating these different approaches for personalization into a single framework, we can exploit the advantages of each approach without necessarily having also to deal with their disadvantages.

4 Software Frameworks for Reuse of Design and Code

> You can't keep everything flexible; some things have to be made of stone. Add flexibility, where you expect that flexibility comes; set things static, where you expect that nothing changes.
>
> Bertrand Meyer at the 2nd Int. Workshop on Practical Problems of Programming in the Large, Glasgow, Scotland, 2005

Targeting at a software framework for providing generic support for personalized multimedia applications raises the question of a proper software engineering support to develop such a framework. A software framework typically constitutes a semi-finished software architecture [IEE00] for a complex application domain [SGM02] that can be adapted to the needs and requirements of a concrete application in the domain. This reuse of the framework architecture in different concrete applications enables the software framework to serve for the important and challenging issue of reuse in the large [DW99, Gla04]. In addition to the architectural reuse, a software framework typically also provides for reuse of its code by the concrete applications.

The motivation for developing a software framework is often based on very good expertise in a specific domain and the possibility to sell several similar software products sharing common code. The knowledge application developers have gained by developing applications in a specific domain for years is abstracted into a software framework to ease reuse and to define a common architecture for the domain. Employing a framework is especially useful in those cases, where flexible but easy to reuse software architectures need to be developed [Pre97d]. The effort for developing a framework is significantly higher than with the conventional application development. However, in return the effort needed for developing the single concrete applications will be significantly reduced (cf. [RJ97]). Consequently, developing a software framework must be considered as long-term investment that returns the effort only if the framework is applied within multiple applications [Pre97b].

In the following Section 4.1, we introduce the reader into the notions and issues of software frameworks. We present the characteristics of software frameworks and introduce the two kinds of object-oriented frameworks and component frameworks. Frameworks do not emerge overnight but are the results of a longer development process. Hence, in Section 4.2 some general issues in regard of the development process of software frameworks are considered and todays support for developing object-oriented frameworks and component frameworks is presented. In particular, the well-known and proved hot-spot-driven design process for object-oriented frameworks by Wolfgang Pree is described and today's lack of a proper software engineering support for developing component frameworks is discussed. Based on this, we finally draw some conclusions for our ProMoCF approach for developing component frameworks presented in Section 7.

4.1 Notions and Issues of Software Frameworks

In this section, we introduce the notions and issues of software frameworks. In Section 4.1.1, the general characteristics of frameworks are presented. Basing on this, the term of framework is clarified in Section 4.1.2. Here, we distinguish between object-oriented frameworks and component frameworks.

4.1.1 Characteristics of Software Frameworks

The central characteristics of software frameworks are the reversal of control flow, the definition of a concrete architecture for a specific domain, and the adaptability of this architecture through predefined variation points. These characteristics are briefly discussed in the following.

Inversion of Control Flow Traditionally developed software applications typically use a set of (object-oriented) class libraries to realize their own functionality. The control flow of such applications, i. e., the order of executing the operations [BD00], is controlled by themselves. The class libraries are merely called by the application and never possess the main control of the application. This calling of external class libraries is called call-down-principle and is depicted in Figure 4.1(a). The upper part of the figure shows the concrete application and the lower part depicts the different functionalties that are provided by an external class library. Whenever the application needs some functionality of the class library, it calls the functionality provided by the library. A class library typically does not define a concrete application architecture, i. e., application developers have to design the architecture of their application by themselves.

Software frameworks however invert the application's flow of control [Has02]. In contrast to conventionally developed applications, the main control of an application developed by using a software framework is governed by the framework [Ack96]. This mechanism of calling is called call-back-principle or Hollywood-principle ("Don't call us, we'll call you!" [1]) and is depicted in Figure 4.1(b). The corpus of the application architecture is already defined by the framework. The

[1] Enervated saying of film producers to get rid of permanently calling starlets [Ack96].

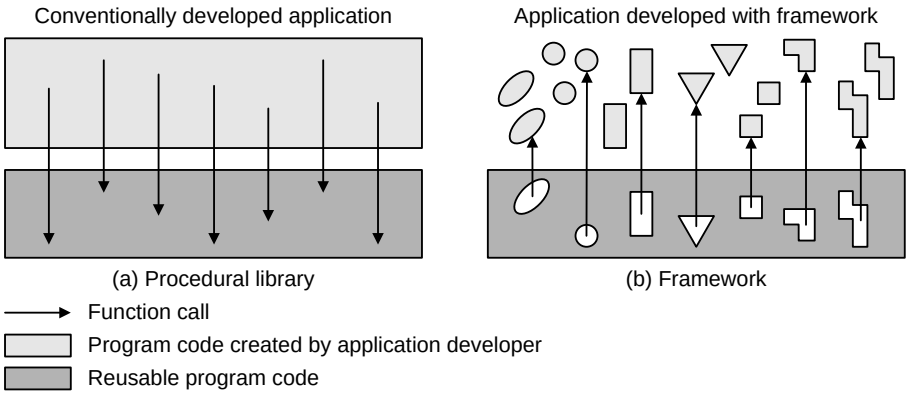


Figure 4.1: Depiction of the (a) call-down- and (b) call-back-principle (from [Ack96, FPR02])

application-specific functionality of a concrete application that is using the framework is called from within the framework. Consequently, the framework possesses the main control of the application [JF88].

Definition of a Concrete Application Architecture Well designed frameworks already define the corpus of the concrete application architecture [Pre97d] and keep it flexible only at specific points. At these points—and only there—application-specific functionality can be integrated into the framework (see also the following paragraph and, e. g., [FPR02, Ack96]). The reversal of control allows software frameworks to serve as generic, extensible, semi-finished software architectures ("skeletons") for concrete applications in the domain. The functionality that is added by the application developers adapts the generic algorithm that is defined within and implemented by the framework to the needs and requirements of a specific application [JF88]. The concrete software architecture defined by a framework can also be considered as a standard architecture for the framework's domain.

Adaptation by Variation Points To serve for a multitude of applications in a specific domain, software frameworks provide flexible variation points at which the framework can be adapted and extended to the requirements of a concrete application. To be more precisely, these variation points are so-called open variation points, since the number of concrete instances for these variation points is not specified by the framework and can be extended by new instances when developing concrete applications. Variation points are those parts in the framework architecture, where the decision for a specific functionality is already defined or typed to several options. However, the actual implementation of this functionality is left open and will be determined by the concrete applications (cf. [CBB+03]).

4.1.2 Types of Software Frameworks

Historically grown, there exist two kinds of software frameworks, the object-oriented frameworks and the component frameworks. They differ in the technology used for adapting to the requirements of a concrete application: In case of object-oriented frameworks, the adaptation and specialization for a concrete application is conducted by overriding abstract classes or by different implementations of predefined interfaces. With component frameworks, specialization only takes place by composing different software components that implement the component framework's interfaces. In addition, also hybrid frameworks exist, which use both technologies for adaptation.

4.1.2.1 Object-oriented Frameworks

Since the introduction of object-oriented frameworks in the late eighties, they have gained a lot of attention. Today, object-oriented frameworks exist for a variety of application domains [Bos00, BMM+99]. In regard of defining the concept of object-oriented frameworks most authors agree that an object-oriented framework is a reusable software architecture. The reuse of a framework includes both reuse of design as well as program code [BMM+99, Ack96]. Apart from this congruence many definitions differ. For example, in [Bos00] object-oriented frameworks are divided into smaller constituents and in [Gov99] some specific subtypes of frameworks are defined. According to Bosch [BMM+99] the probably most cited definition of object-oriented frameworks is the one by Johnson and Foote in [JF88] describing an object-oriented framework as a set of classes defining an abstract solution for a family of similar problems. This set comprises concrete and—in particular—abstract classes [WBJ90] providing an incomplete design and implementation for applications in a specific domain [Bos00, Pre95a]. Consequently, a software framework constitutes an incomplete design and an incomplete implementation for an application in a specific domain and problem area, respectively [Bos00]. The variation points of object-oriented frameworks are the abstract framework classes (with the white-box object-oriented frameworks) and the framework interfaces (with the black-box object-oriented frameworks), respectively. Many authors follow this definition, e. g., [SGM02], [GHJV04], [BD00], [BMR+96], [Ack96], and [Pre95a]. The object-oriented framework defines the structure of the classes and objects as well as their responsibilities. It determines the collaboration of the classes and objects as well as the control flow [GHJV04, Pre95a]. An object-oriented framework predetermines the architecture of an application such that the application developers can concentrate on the details of the application. The application developers extend the object-oriented framework with application-specific functionality by defining concrete subclasses of the abstract framework classes [Bos00]. These self-defined subclasses are called by the framework following the call-back-principle (see Section 4.1.1).

The Open/Close-Principle An object-oriented framework should meet the open/-close principle [Zül05, Mey97]. This means, it shall be open in regard of (future) extensions and closed concerning modifications. Open in regard of (future) extensions means that the object-oriented framework can be extended by new functionality, even such functionality that has not been predicable when designing the frame-

work. Closed in regard of modifications means that it shall not be able to change the abstract application logic provided by an object-oriented framework by inheriting concrete subclasses in order to obtain a different application behavior than intended by the framework developers. As a consequence, the concrete subclasses only implement the abstract methods defined in the super-classes of the object-oriented framework. The concrete classes and methods already implemented in the framework shall not be overridden or modified. Otherwise, inheritance would break the encapsulation of the object-oriented frameworks [WS96]. This can be achieved, e. g., in the programming language Java by using final-methods [Sun06].

Reuse of Object-oriented Frameworks The behavior of an object-oriented framework can be adapted to the requirements of a concrete application in two different ways [JF88, BD00]. These are provided by the white-box reuse and the black-box reuse of frameworks.

With white-box reuse or white-box frameworks [BD00], the behavior of an object-oriented framework is adapted by inheriting concrete classes from the abstract framework classes. Here, the abstract methods of the framework class are implemented in the concrete subclasses of the application [JF88]. Parts of the implementation of the white-box framework are open for inspection, however, without allowing for any modifications in the internal implementation (cf. [SGM02, p. 555]).

In contrast to white-box frameworks, the adaptation mechanism for black-box reuse or black-box frameworks does not base on abstract classes but on a set of prefabricated, i. e., concrete classes that implement the framework's interfaces. With black-box frameworks, reuse takes exclusively place on basis of well-defined interfaces of the framework classes [BD00] and their contractual specification (cf. design-by-contract in [Mey92]). This means that changing the framework's behavior is realized by different compositions of concrete classes and not by inheritance. For it, application developers implement new, application-specific functionality against the external framework interfaces. Consequently, the quality of the contractual specification is crucial with black-box reuse [Pre97d].

By partially opening a black-box-framework towards white-box usage "arbitrary shades of gray" can be build [SGM02]. Actually, most object-oriented frameworks are neither pure black-box nor pure white-box ones. However, white-box frameworks typically mature to a black-box framework when often reused [RJ97, JF88]. Typically, this takes place with numerous reuse of a white-box framework within concrete applications. Some authors further distinguish between white-box frameworks and glass-box frameworks (see [SGM02, 40ff.]). While white-box frameworks allow for manipulation of the implementation, glass-box frameworks merely allow for studying it.

Composition of Object-oriented Frameworks Originally, it was assumed that for the construction of an application only one object-oriented framework will be employed. However, in the last couple of years one can observe that increasingly several frameworks are used for the construction of an application [SGM02, Bos00, Mat00, MB97]. So in most of today's fully functional software systems multiple software frameworks are employed [SGM02]. The reason for this lies in the fact that today's applications more and more need to embrace multiple domains. An object-

oriented framework, however, is only designed for one application domain. Using multiple, independently developed object-oriented frameworks for the construction of a single application often leads to crucial framework integration problems (see, e. g., [BMM+99, Pre96b]). Due to the nature of object-oriented frameworks, each framework intends to overtake the main control flow of the application [SGM02]. The reason for this lies in the inversion of the control flow (see Section 4.1.1). In addition, the functionality of the employed object-oriented frameworks can overlap [BMM+99]. Here, specific parts of the considered application domain are modeled and covered by multiple frameworks. This often also concerns the base classes of the object-oriented frameworks. As a consequence, one tries to make the object-oriented frameworks compatible by adapting and modifying the base classes of the frameworks. However, this adaptation process can result in requiring numerous modifications in the derived classes. This can lead to discard fundamental design decisions of the object-oriented frameworks [Pre96b]. The problem that occurs when a class and its subclasses are evolving independently is called the fragile base class problem [SGM02, WS96].

The interaction between different object-oriented frameworks can only be managed on a higher abstraction level. For it, a level of using object-oriented frameworks is necessary which describes where, when, and how parts of the frameworks overlap or interact [SGM02]. The idea here is to develop so-called frameworks of second order. In the way an object-oriented framework uses abstract classes and interfaces for combining sets of concrete classes, frameworks of second order could be used to combine abstract subsystems that are realized by frameworks of first order. Like a regular object-oriented framework supports the combination of concrete classes, high-order subsystem frameworks—the frameworks of second order—could serve for combining more complex subsystems. Each of these subsystems could be structured and implemented as traditional object-oriented framework (of first order) [SGM02]. Efforts towards this direction are component frameworks, which are presented in the following section.

4.1.2.2 Component Frameworks

When considering component frameworks, we first need to introduce and define the concept of software components. In literature, many different definitions of software components can be found. What a software component is and what it is not is still subject of dispute, e. g., whether a single object can be a software component or not [SGM02].

However, in recent years some characteristic properties of software components have been reinforced. These characteristic properties are [SGM02, Szy00]:

- A software component is a unit of independent deployment.
- A software component is a unit of third party composition.
- A software component has no (externally) observable state.[2]

Following these characteristics, it is essential that a software component encapsulates its implementation and interacts with the environment by means of well-defined interfaces [SGM02]. Therefore, a software component needs to come with a

[2] This means that a software component cannot be distinguished from copies of its own.

clear specification of what it requires and provides. Based on these principal characteristics, software components are defined within this work as modular units of a software system that encapsulate their content and thus their internal (complex) behavior. They appear to the environment as exchangeable units and interact with the environment via well-defined (contractually specified) interfaces.

Based on the definition of software components, we can now consider and determine the concept of component frameworks: While object-oriented frameworks are defined by the structure of their classes and objects and the responsibilities between them, the focus of component-based frameworks or component frameworks lies on the single software components and their interfaces. Consequently, component frameworks are defined as a collection of different components with a predefined cooperation behavior and the aim to fulfill tasks in a specific domain [Pre97b]. Szyperski et al. state in [SGM02] more precisely that a component framework—there denominated as contextual framework—is a collection of rules and interfaces, which control the interaction of components within a component framework (q. v. [Wec97, FSJ99b]). These rules and interfaces constitute the contractual-like agreements between the provider of a "service" (the software component) and its "user" (the component framework and the concrete application, respectively). These contractual agreements can comprise besides functional aspects also non-functional (or extra-functional [RS02]) aspects of the services. As a component framework is designed to provide "standardized" support for a specific domain [Bos00], it is also called a domain-specific component framework.

A component framework allows to combine independently developed instances of the framework components [SGM02]. These component instances, which implement the framework component's interfaces and follow the component's contract, are plugged in at specific extension points and executed by the framework. These extension points are defined and typed by the framework components' contracts. The component framework controls the interaction between the instances of the components [Wec97]. Consequently, the main control flow of the application resides—like with the object-oriented framework—at the component framework. The single components of a component framework do not provide for a concrete application architecture. However, the component framework itself, i. e., the organization and the structure of the single components emerged from the requirements of the considered application domain does. In contrast to object-oriented frameworks, adapting the component framework's behavior to the requirements of a concrete application is only allowed by composition. However, instead of single concrete classes, different instances of the framework components (and their classes) are composed. Consequently, the components constitute the variation points of a component framework.

Frequently, the single components of a component framework are developed by means of object-oriented frameworks [Bos00]. Here, only one object-oriented framework should be employed for the development of exactly one framework component. In addition, the object-oriented framework should not be divided over multiple components in order to avoid class inheritance relationships between components. Such an inheritance breaks encapsulation and thus is considered harmful [SGM02, WS96]. One of the most important characteristics of software components is violated, namely the issue that a component shall be independently deployable (see Section 4.1.2.2).

A component framework can be employed alone or cooperate with other components or component frameworks. In the first case, the application is determined by the single component framework. To be able to cooperate with other component frameworks, they should be modeled as software components. The concept of component frameworks can also be applied hierarchically. Component frameworks that are realized as components are—in contrast to object-oriented frameworks—designed such that they can cooperate with other component frameworks. Thus, component frameworks are more easier to integrate [Sih01]. Component frameworks are currently considered as the highest level of software architecture reuse [Bos00].

4.1.2.3 Hybrid Software Frameworks

Besides the presented kinds of software frameworks also hybrid forms are possible. Hybrid frameworks possess characteristics of both object-oriented frameworks and component frameworks. This means that one part of the framework is adapted to the requirements of a concrete application by inheritance while another part is conducted by implementing interfaces and composing concrete classes and components. By shifting the proportion of the parts adapted by inheritance and composition any variants of hybrid forms are possible.

4.2 Development of Software Frameworks

In the previous section, we introduced the concepts of object-oriented frameworks and component frameworks. Now, we discuss in Section 4.2.1 some general issues in regard of developing software frameworks. In Section 4.2.2, we present todays support for developing object-oriented frameworks. Here, the well-known and proved hot-spot-driven approach for designing object-oriented frameworks is presented. In Section 4.2.3, we discuss the lack of an appropriate support for developing component frameworks today.

4.2.1 Development Process for Software Frameworks

Due to the complexity of frameworks and the frequent changes of the framework's flexibility requirements [BMM+99] is the development of software frameworks characterized by an iterative and incremental development process. Frameworks are developed in many small iterations. The architecture and interfaces of the framework have to be redesigned again and again. Beginning with some example applications in the considered domain, relevant functionality is gradually abstracted and integrated into the framework. During the analysis of the application domain, it has to kept in mind that the behavior and the configuration of the framework can change for different concrete applications. On account of this, a process model for framework development should provide an explicit activity for identifying and specifying the framework's variation points, i. e., those parts of the framework, where application-specific functionality can be integrated (cf. [CBB+03]). This activity should be supported by an appropriate framework development method.

The development of an object-oriented framework as well as component framework is a challenging task. It is profitable only, when multiple applications are developed with this framework. From experience it is known that at least three applications are necessary to gain a return of investment of the higher development efforts of a framework compared to the development of traditional applications. A high-quality software framework is typically the result of a long development process. Hence, software frameworks should be developed by using a proper process model supported by a corresponding development method. This is essential to building a successful framework [CC02]. The process model describes which activities for developing the framework have when to be conducted. Such a process model needs to provide support for the framework developers to consistently create the artifacts that the framework consumer, i. e., the application developers need [CC02]. The corresponding development method supports the framework developers in how to conduct the process model's activities for creating the artifacts.

The development process of frameworks is characterized by an iterative and incremental procedure due to changing requirements and the complexity of frameworks (cf. [BMM+99]). The relevant framework functionality is abstracted from example applications and other existing software products in the domain by stepwise abstraction. This includes structuring the framework in regard of the supported adaptation possibilities, i. e., the provision of flexible variation points to adapt and extend the framework. These variation points can be identified and refined in the iterative framework development process [JN99]. The experience gained with applying the framework for the development of concrete applications are the input for the next iteration of the framework development process. This iterative development of frameworks can be coarsely divided into the following phases (see [BMM+99]): *development*, *usage*, *composition* as well as *evolution and maintenance* of frameworks. The characteristics and challenges of these phases are presented in the following sections.

Development of Frameworks The development of frameworks differs very much from the conventional development of applications. The main reason for it lies in the fact that a framework needs to consider and support all concepts of a specific domain. However, with the development of a particular application only those concepts are of interest that are necessary to realize this specific application. The development of frameworks comprises those activities that are typically to be found with process models for the traditional software development (see [BMM+99]): analysis of application domain, definition of an architecture, design, implementation, test, and documentation of the framework. However, the development methodologies for die traditional application development do not sufficiently support the design and implementation of frameworks. The design of frameworks requires some new concepts that need methodological support [BMM+99]. For example, during the analysis of the application domain, one needs to take into consideration that in contrast to the regular application development the behavior and configuration of a framework can change in different applications. In addition, a requirements analysis for the framework is typically not feasible, as the concrete functionality of the framework is not known in advance and only emerges with iteratively refining and evolving the framework.

With developing a framework, appropriately determining the considered domain scope is crucial. If the domain of the framework is selected to large, then the development team will not have the necessary experience and knowledge in this domain in order to develop a framework for it. This can result in higher development costs as well as restrict the applicability and usability of the framework. The development of a small framework is financially less risky and easier to conduct. However, here there is the peril that the framework is susceptible to changes of the application domain [BMM+99]. Applications that use such a small framework can quickly grow up above its limits [BMM+99] and adding new functionality is costly.

Usage of Frameworks Using a framework means to apply and adapt it for the development of a concrete application. Learning how to use a framework possibly takes a lot of time [BMM+99] and is directly dependent to the size and number of the provided variation points.

A fundamental prerequisite for the successful application of frameworks is good documentation. Here, so-called framework cookbooks are a good means to support application developers with applying and adapting a framework to the requirements of a concrete application [Pre97e, Pre95b]. The "recipes" of these "cookbooks" describe in an informal way the single steps that are necessary to conduct typical tasks in order to applying and adapting the framework. However, these recipes do typically not explain the internal design and implementation details of the framework. Besides a framework cookbook, another possibility to learn the usage of a framework is to obtain support by a mentor. This mentor was ideally involved in the development of the framework.

Composition of Frameworks With the composition of frameworks, multiple frameworks are applied for the development of a concrete application. Here, the employed frameworks need to be adapted to the architecture of the application as well as to the other frameworks, in order to be able to collaborate. The composition of frameworks can lead to enormous problems, since frameworks by their nature endeavor to gain full control about the application (see Section 4.1.1). In extreme case, the architectural styles [SG96] of the frameworks are so different, that it is unlikely to compose them. This incompatibility of architectures is termed architectural mismatch [GAO95]. A first step for solving this problem seems to be to explicitly specify the architectural style underlying a framework [BMM+99]. If for the development of an application multiple frameworks are applied, it can happen that the frameworks partially overlap in their provided functionality. Here, specific parts of the application are modeled in multiple frameworks, each from its own point of view. Consequently, these different views need to be brought together (see [BMM+99, p. 73]).

Evolution and Maintenance of Frameworks The development of software frameworks needs to be considered as a long-term investment. As such, software frameworks should be advanced and maintained accordingly. With the evolution and maintenance of software frameworks is understood the modification and adaptation of frameworks to changing requirements of the employing applications and the considered domain (cf. software evolution and maintanence in [Som04]). The evolution

and maintenance of a framework begins already during the development and not subsequent to the first release of the framework (cf. [Som04, 82ff., 488ff.]). Already in the early development of a software framework many iterations are necessary for designing the framework. The reason for it lies in the goal of frameworks to provide for reuse. The only possibility to proof this is applying the framework for the development of concrete applications and identifying and improving the weak points of the design (cf. [BMM+99]).

For a successful development of frameworks, not only software engineering knowledge is important. However, also organizational aspects must be taken into account (cf. [RE99]). The development of frameworks requires a change in the organization structure of the developer team. The team should be divided into one part concerned with the development of the framework and a second part that applies and evaluates the developed framework by employing it for the development of concrete applications in the domain. This is a consequence of Weinberg's law [ER03] saying that a developer is unsuited to test his or her code.

4.2.2 Development of Object-Oriented Frameworks

Designing an object-oriented framework requires a lot of experience and experimentation [JF88]. A well designed object-oriented framework is typically the result of many design iterations and a lot of hard work [WBJ90]. The strive for adding more and more flexibility does not automatically lead to a high-quality framework. Rather, it is important to deliberately implement flexibility into the framework reflecting the requirements of the considered domain in a reasonable manner [Pre97d]. Actually, adding unnecessary flexibility noticeably increases the complexity of the framework [FPR02].

The hot-spot-driven approach by Pree is a well-known and proved process model with corresponding development method for designing object-oriented frameworks. It is multiple published in literature, e. g., [FPR02, Pre99, Pre97c, Pre96a, Pre95a]. As we will see in Section 7, this process model is employed as basis for our approach to improve the development process of component frameworks. Therefore, this process model is presented in detail in Section 4.2.2.1. In addition, other design methods for object-oriented frameworks are briefly presented in Section 4.2.2.2.

4.2.2.1 Pree's Hot-spot-driven Approach

When designing a framework, the challenge is to identify the points where the framework should be flexible, i. e., to identify the semantic aspects of the framework's application domain that have to be kept flexible. These points are the so-called hot-spots and represent points or sockets of the intended flexibility of a framework [Pre95a]. Since the quality of an object-oriented framework is—as described in the introduction of Section 4.2.2—highly dependent on the flexibility characteristics in regard of the considered domain, there is an explicit activity in the hot-spot-driven design process for identifying variation points [FPR02]. A carefully conducted identification of variation points, the so-called hot-spots, can be a valuable support for developing good frameworks [Pre97d]. The single activities of the hot-spot-driven design process are described in the following. As notation for the class

diagrams, we employ UML-F [FPR02, FPR00], an extension of the Unified Modeling Language (UML) [OMG05] for describing frameworks.

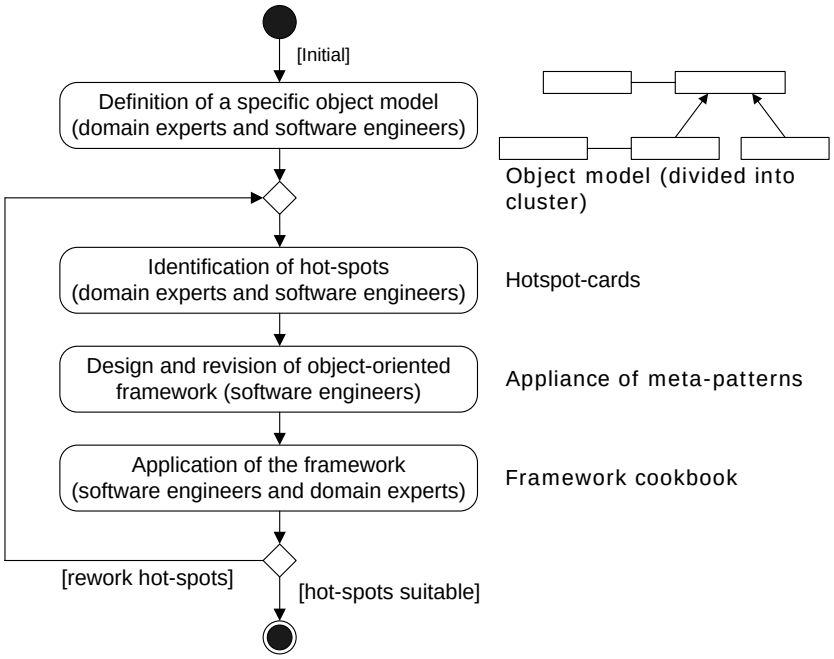


Figure 4.2: The hot-spot-driven design of object-oriented frameworks [FPR02, Pre95a, Pre97d])

Definition of a Specific Object Model The critical first step with the development of an object-oriented framework is the identification of the main abstractions of the considered application domain [FPR02], the so-called key abstractions or archetypes [BB99, LBHB99]. For object modeling chiefly application-specific knowledge is required. The computer scientists support the domain experts with conducting this activity. However, the differentiation between the both roles is only hypothetical and shall express that different kind of knowledge is necessary for conducting this activity.

The initial object model forms the basis of the hot-spot-driven framework approach. For creating this object model, e. g., CRC-cards (Class Responsibility Collaboration cards) can be used [FPR02]. CRC-cards [BC89] are generally suited for identifying classes, objects, and their associations [Pre97d]. Besides the CRC-cards it is recommended to create a separate set of cards that capture only the abstract classes of the developed object-oriented framework. These are referred as aCiRC-cards (abstract Class/interface Responsibility Collaboration cards). The initial object model is divided into several clusters [Mey90] on basis of the identified key abstractions. A cluster is a group of classes and/or interfaces. The framework development process is divided according to the clusters into several, chronologically overlapping and independent software life cycles. The single clusters can be in dif-

ferent development phases, independent of the phase in which the hot-spot-driven design of the developed framework currently is. This procedure reflects the fact that for an iterative design of abstract classes and interfaces some of the clusters need be in the implementation phase. For example, if during the implementation of a cluster is realized that one abstraction of another cluster is not appropriate, redesign of this abstraction needs to be conducted in parallel with the implementation of the other cluster [FPR02].

Before the main cycle of the process is conducted, it is helpful to have several object models of similar applications in the considered domain. These object models allow for an easier identification of common features and their abstraction by the framework. However, such object models are typically not available, as the development of an application-specific software solution itself is a complex and iterative task in which object models need to be created and refined.

Identification of Hot-spots and Writing of Hot-spot-cards The main problem with identifying hot-spots is that application experts are not used to abstract and generalize from a concrete software system. Typically, application experts are not familiar with object-oriented concepts such as objects, classes, inheritance, patterns, and frameworks. Instead, they are good in modeling software functionalities. They typically also know what the fundamental elements of their application area are. Consequently, a common basis for communication between the computer scientists and the domain experts needs to be established. This communication needs to base on the functionality of the framework and does not need to be concerned with placing the functionality into the framework classes and interfaces [FPR02, Pre97d]. One possibility to communicate the framework functionality are hot-spot-cards or variation point cards. Hot-spot-cards are another variant of the CRC-cards. They provide a simple but effective means for documenting and communicating the flexibility requirements between the domain experts and the framework developers [Pre97d]. In a first step, they are used to identify the hot-spots of the framework. In a second step, they support designing the framework.

In principle, functionality as well as data can be identified as hot-spots. However, most hot-spots will probably be functionality [Pre97d]. Depending on the type of the identified hot-spot, either a function- or data-hot-spot-card is created. The structure of these cards is depicted in Figure 4.3. Function-hot-spot-cards document the functionality that shall be kept flexible with the object-oriented framework. With data-hot-spot-cards the elements of the application domain are captured that need to be generalized in the course of the framework development. For concrete examples of function- and data-hot-spot-cards, we refer to the literature, e. g., [Pre96a].

For implementing a data-hot-spot-card, typically an abstract class is added to the object model of the framework. Consequently, the documentation of data-hot-spots by corresponding cards can be omitted and the object model of the framework can be directly manipulated [Pre96a]. For that reason, we will concentrate in the course of this work on realizing the function-hot-spots of a framework. In addition, for the remainder of this work we agree that with a hot-spot(-card) implicitly a function-hot-spot(-card) is meant.

For transforming the object model by means of hot-spot-cards into an object-oriented framework, the granularity of hot-spots should be *exactly one* method in

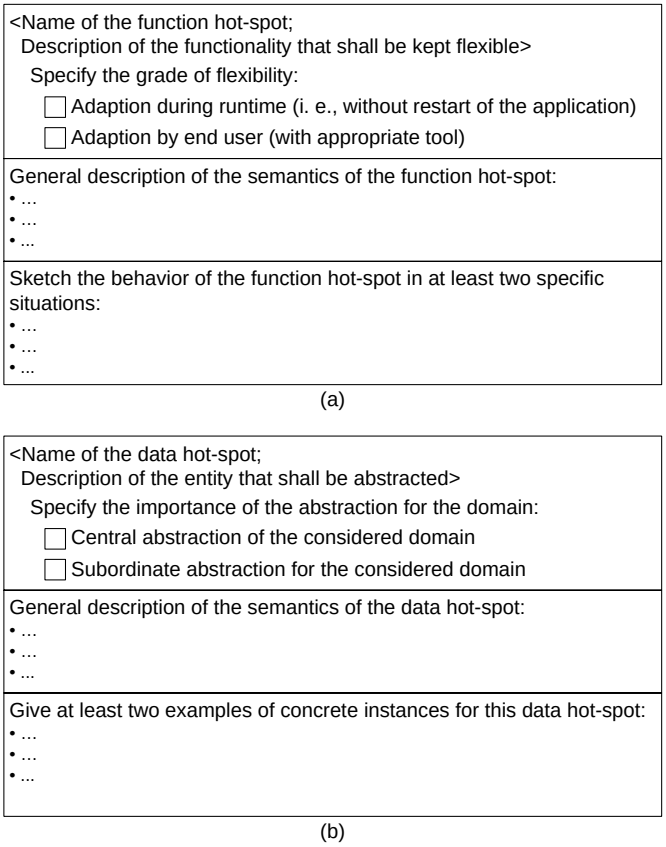


Figure 4.3: Structure of (a) function-hot-spot-cards and (b) data-hot-spot-cards (based on [Pre96a, Pre97d])

the programming language. This recommendation deviates from the literature for the hot-spot-driven design of object-oriented frameworks [Pre95a, Pre97d], which defines the granularity of hot-spot-cards to represent *approximately one* method. The reason for this deviation defining the granularity of hot-spots to be exactly one method is supported with the introduction of so-called group-hot-spot-cards in Section 7. These group-hot-spot-cards are used to identify the components of a component framework and to define the flexibility requirements to these components.

(Re-)Design of the Framework with Meta Patterns The initial hot-spots are identified and written down on hot-spot-cards. Now, the object model is modified by the computer scientists according to the flexibility requirements recorded on the hot-spot-cards. For each hot-spot that is described on the hot-spot-cards, a distinct hook is defined in the design of the object-oriented framework, i. e., the computer scientists define where to place the methods—that belong to the hot-spot-cards—in the class hierarchy of the framework [FHLS99]. These methods are called hook methods. Framework-centric construction patterns, the so-called meta patterns,

support the computer scientists in the concrete implementation of the hot-spot-cards [Pre95a]. Meta patterns constitute essential construction patterns for object-oriented frameworks [Pre97d] and describe how to design a framework independent of a specific domain [Pre95a]. They are formed of template methods and the already mentioned hook methods [Pre95a]. The hook methods constitute the abstract placeholder in the object-oriented framework that are called by the already implemented template methods of the framework. Template methods can describe abstract behavior, generic control flow, or the interaction behavior of objects in the framework. The principle idea of hook methods is that the behavior of the object-oriented framework is modified by the application developers by overriding the hook methods [BD00]. A specialization, i. e., adaptation of the object-oriented framework takes exclusively place by the framework's hook methods [Pre95a, Pre97d]. This ensures the observance of the open/close principle (see Section 4.1.2.1). Template methods describe the abstract application logic of the object-oriented framework that must not be modified for further adaptations and are thus also called frozen-spots.

Figure 4.4 presents an example of the concept of template methods and hook methods. The template method t() of class A calls—as depicted in Figure 4.4(a)—the hook method h(). The class B in Figure 4.4(b) overrides the hook method h() by inheriting from class A. The corresponding UML class diagram is depicted in Figure 4.4(c).

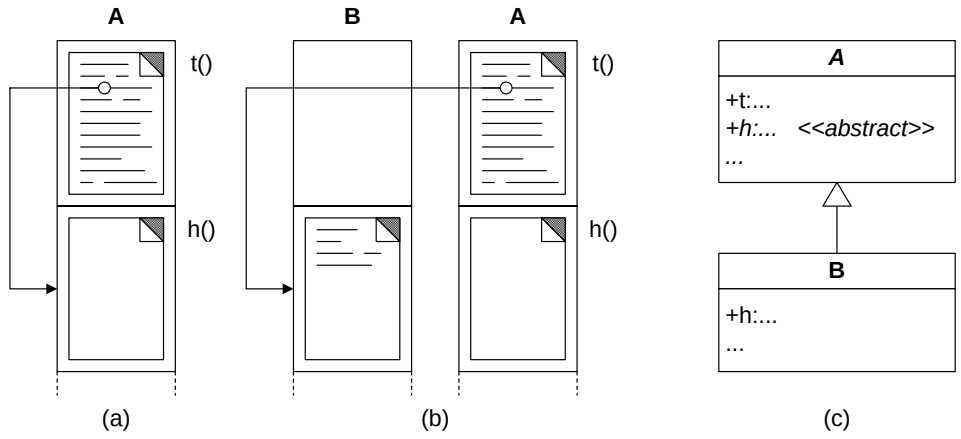


Figure 4.4: (a) Template and hook methods, (b) Overriding of hook methods, and (c) the corresponding UML class diagram (based on [Pre97d])

As depicted in Figure 4.5, template and hook methods can be defined either in a common class or in two separate classes. A class that contains hook methods is called a hook class and a class that contains template methods is called template class. Thereby, the hook class parameterizes the template class. The class A in Figure 4.4(c) contains template methods as well as hook methods. It is both, template class and hook class and parameterizes itself. The UML-F [FPR02, FPR00] provides two tags to make template and hook methods in the design of an object-oriented framework explicitly recognizable. As depicted in Figure 4.5, hook methods are pro-

vided with the tag «hook» and template methods have a «template» tag. These tags are also used to label hook and template classes.

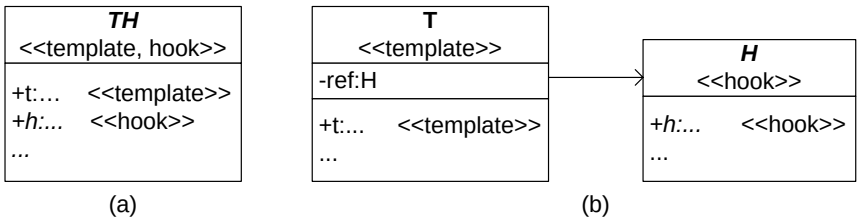


Figure 4.5: (a) Unification and (b) separation of template and hook methods [Pre99]

Defining the hook and template methods in a common class or in two separate classes depends on whether adaptation during runtime shall be supported or not. Here, either the unification principle or the separation principle is applied [Pre97d]. As depicted in Figure 4.5(a), the template and hook methods are defined following the unification principle in a common class TH. Here, adaptations can be conducted by creating corresponding subclasses and requiring a restart of the application. To provide for adaptation during runtime, the hook and template methods are divided according to the separation principle as depicted in Figure 4.5(b) into two classes. The behavior of the template class T can be modified by compositions with different H-objects, i. e., instances of the concrete classes that implement H. The directed association between T and H expresses that a T-object refers to an H-object. This is typically realized by an (instance) variable in the template class T, here ref. Consequently, the H-objects can be exchanged during runtime without restarting the application. It is important to note that the abstract class H can also be an interface, e. g., in the programming language Java. In principle, the realization by an abstract class is on a par with an interface [PAS98]. However, in case of doubt whether adaptation during runtime is necessary, using an interface is to be preferred (cf. [RJ97]). In addition, a configuration tool needs to be developed when an adaptation by the end users shall be provided.

Besides the meta patterns presented so far, the template and hook classes can also be combined recursively. As depicted in Figure 4.6(a), the template class T can be a subclass of the hook class H. In the extreme case, template and hook class coincide and are merged into a single class like it is depicted in Figure 4.6(b). The template class T can—as subclass of H—manage another instance of itself as an implementation of the hook class H. However, the reverse is not possible. The abstract hook classes cannot be subclasses of concrete template classes. This recursive nesting is not only of theoretical interest but also relevant for practice as examples show, e. g., in [Pre95a, Pre96a]. In addition, the meta patterns exist also in another variant where the template class not only manages *one* instance of the hook class but carries a list of object references to manage an arbitrary set of hook classes (see [Pre95a, Pre94]).

The design of the classes' interfaces generally requires multiple iterations. This is primarily caused by the different use of an object-oriented framework in different concrete applications. During one of these iterations it can emerge that the template

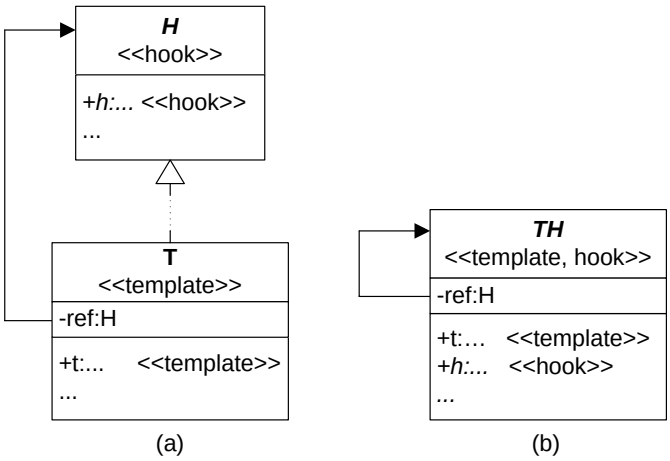


Figure 4.6: Recursive combination of template and hook classes (from [Pre99])

methods are to rigid and thus the framework is becoming inflexible. Consequently, new hook methods need to be defined and added to the framework. On the other hand, new template methods can emerge by a repetitive application of the object-oriented framework. For example, if several application developers implement a similar control flow when adapting the framework, this control flow should possibly be added as template method to the framework.

Usage of the Object-oriented Framework An object-oriented framework needs to be reused several times, i. e., it needs to be employed by concrete applications, in order to identify the weak points of its design (cf. [BMM+99, p. 63]). The weak points of the design of an object-oriented framework are those where the hot-spots are inappropriate. This means that either hook methods are missing or providing an inappropriate abstraction. Weak points can also be template methods encapsulating too much or less (abstract) behavior. The cycle depicted in Figure 4.2 constitutes an evolutionary process for designing object-oriented frameworks by the means of hot-spot-cards. The explicit identification of hot-spots provides for a significant reduction of iterations needed for framework development.

4.2.2.2 Other Design Methods

In the previous section, we presented in detail the hot-spot-driven design process for object-oriented frameworks. Besides this approach, there exist also other approaches in literature. For example, the framework design by systematic generalization as described in [Sch99], the application of templates for specifying frameworks proposed in [DW99], a role-based approach for developing object-oriented frameworks presented in [Rie00b, RG98], and a process model with a comprehensive guideline comprising 71 steps for the framework development is described in [LN95]. A survey of methods for developing and applying object-oriented frameworks has been conducted by Mattsson [Mat96]. In his further work, he evaluated,

improved, and validated methods for the evolution of object-oriented frameworks [Mat00].

4.2.3 Development of Component Frameworks

For the development of component-based software, there exist many process models like [Som04, CJC02, Bos00, CD01]. Most of these process models define the following phases: identify components, specify components, search for existing components, design components, implement components, test components, and finally use the components. To adequately support these process models, an appropriate method is needed, e. g., for identifying and specifying components.

In contrast to object-oriented frameworks, for which we find with the hot-spot-driven approach an elaborated process model, a proper approach for developing component frameworks is still missing. Currently, the development of component frameworks is conducted *ad hoc*. Consequently, the quality of component frameworks is eventually dependent on the experience and skills of the framework developers [Sih01].

To improve the development of component frameworks, the design of software components could be directed in such a way that flexible and adaptable components are created. This means that software components are to be developed such that they can be easily used to assemble component frameworks. By implementing such flexible components, one hopes that this automatically leads to well designed component frameworks. However, these components are developed without taking the requirements to a concrete component framework in a specific domain into account. Thus, there is the peril of adding more and more flexibility to the components that is not appropriate to the framework domain. However, it is important to deliberately implement flexibility into a component framework reflecting the requirements of the considered domain in a reasonable manner [Pre97d]. Like with the object-oriented frameworks, flexibility should be well considered when added to the framework. Actually, adding unnecessary flexibility increases the complexity of a framework noticeably [FPR02]. Consequently, the flexibility requirements to a component framework—like with the object-oriented frameworks—should be identified and specified by employing a proper process model and development method.

4.3 Conclusions for the ProMoCF Approach

In the previous sections, we presented the notion and issues of software frameworks. We described the characteristics of frameworks and introduced two kinds of frameworks, the object-oriented frameworks and component frameworks. These two kinds of frameworks differ in the technology used for adapting the framework to the requirements of a concrete application. We presented the elaborated and proved hot-spot-driven design approach for object-oriented frameworks and expounded the problems that arise with composing multiple object-oriented frameworks. To overcome this drawback of object-oriented frameworks and to obtain the benefits of component technology, such as independently deployable and exchangeable components, the development of component frameworks is pursued. However, in contrast to object-oriented frameworks, the development of component frameworks is currently

conducted *ad hoc* and thus dependent on the experience and skills of the framework developers only. What is needed is a systematic software engineering support for the development of component frameworks. This is pursued with the envisioned ProMoCF approach. It is composed of a process model, determining which activity for developing a component framework is when to be conducted. The process model needs to be accompanied by an appropriate development method supporting the framework developers in conducting the activities. The development method needs in particular to provide support for identifying the framework components and specifying the flexibility requirements to these components. Summarizing, the following demands are made on the envisioned ProMoCF approach.

- Provide a process model for the systematic development of component frameworks.
- Provide an appropriate development method for identifying the framework components and specifying the flexibility requirements to these components.

5 The MM4U Approach

Having presented the foundations, the related work, and the challenges of generating personalized multimedia content in the previous sections, we now present our approach for a dynamic authoring of personalized multimedia content. From the existing systems for multimedia personalization presented and analyzed in Section 3.2 and their systematical categorization in Section 3.3, we deduce that a *two-step multimedia content creation approach* is needed to provide for an efficient and economic support for personalized multimedia content. This two-step approach is presented in Section 5.1 and is extended on the basis of the systems analyzed and categorized to a general process chain for multimedia content personalization. From this general process chain for multimedia personalization and the application scenarios introduced in Section 2, we derive in Section 5.2 the requirements to the envisioned MM4U approach. Finally, the layered architecture of the MM4U framework is defined and introduced in Section 5.3, based on the general process chain and the requirements to the MM4U approach.

5.1 The General Multimedia Content Personalization Process

With the MM4U framework, we pursue a two-step approach for creating personalized multimedia content. As depicted in Figure 5.1, in a first step media elements are orchestrated into personalized multimedia content using an abstract multimedia content representation model. In a second step, this content is transformed to the requirements and characteristics of different concrete multimedia presentation formats.

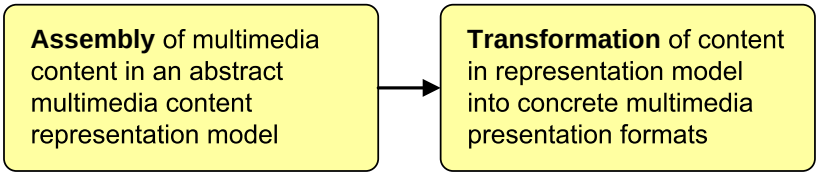


Figure 5.1: A two-step approach for multimedia presentation creation

Extending the two-step multimedia content creation approach, we identified on the basis of the related work the general process of creating personalized multimedia content [SB05c, SB05d]. This process is depicted in Figure 5.2. It involves

different phases and tasks from media selection to the final delivery of the personalized multimedia presentations. We identify the central tasks in this process that need to be supported by a suitable solution for personalized multimedia content creation. As shown in Figure 5.2, the core of this process is an application called "personalization engine". The input parameters to this engine can be divided in three groups:

- Media elements and meta data: The first group of input parameters comprises the media elements with the associated meta data. They constitute the content from which the personalized multimedia presentations are selected and assembled.
- User profile and context information: The second group enfolds information about the user's "personal profile" and "context". The user profiles include information about, e. g., the users' current task, their knowledge, goals, preferences, interests, abilities and disabilities, as well as demographical data. The user context comprises information about the users' location, current time, and environment, like weather and loudness. It also enfolds information about the characteristics of the used (mobile) end device such as the device's hardware and software, e. g., available amount of memory and installed multimedia players, as well as supported network connections and input devices.
- Document structures and layout information: The third group of input parameters influences the structure of the resulting personalized multimedia presentation and subsumes other preferences such as style and layout a user could have for the multimedia presentation. Here, e. g., predefined document structures (templates), rules and constraints, as well as any other layout and style information are embedded to determine among others the temporal course and spatial layout of the personalized multimedia presentation.

Within the personalization engine, these input parameters are used to author the personalized multimedia presentations. Here, the following steps as illustrated in Figure 5.2 are conducted:

Select First, the personalization engine exploits all available information about the user's preferences, needs, situation, and characteristics of the used end device to select by means of media meta data those media elements that are of most relevance for the user and the requested presentation. Result of this step is a personalized selection of media elements. These media elements are used as input for the assembly step.

Assemble In the next step, the selected media elements are assembled and arranged by the personalization engine—again in regard of the user's personal profile and context information—into personalized multimedia content. The personalized multimedia content is represented in an internal multimedia content representation model or internal multimedia document model [SB05c]. This internal representation model abstracts from the syntax and different characteristics of today's multimedia presentation formats. Even though the abstract model does not reflect the fancy features of some of today's multimedia presentation formats, it captures

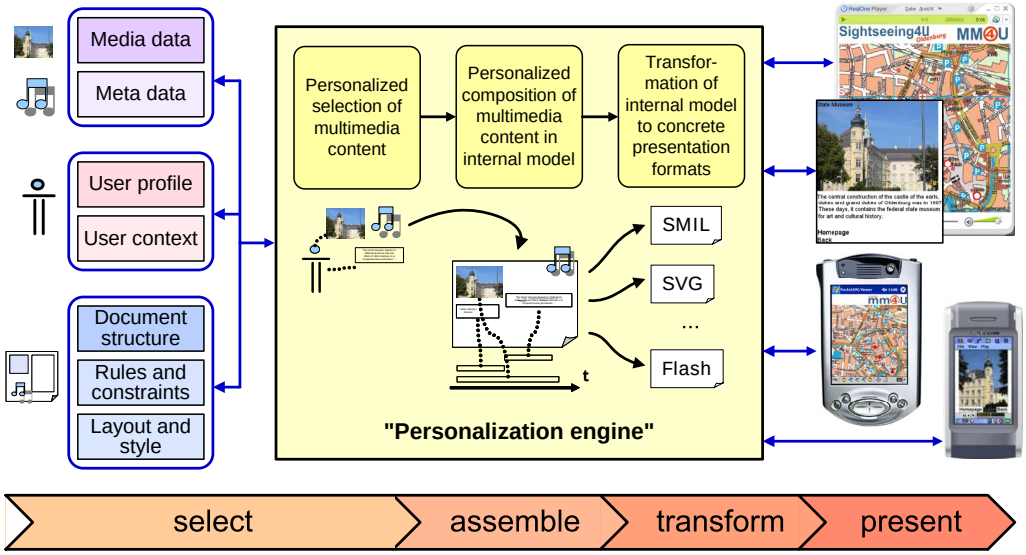


Figure 5.2: The general multimedia content personalization process [SB05c, SB05d]

the very central aspects of multimedia presentations without instantiating these in a concrete syntax and format. These central aspects of multimedia modeling are the definition of the temporal course, the spatial layout, and the interaction possibilities of the multimedia presentation with the users (see Section 3.1.2). The representation model is designed to be efficiently transformed to the concrete syntax and features of the different (standardized) presentation formats, like SMIL, SVG, and Flash. This enables our approach to serve different target multimedia presentation formats and reach multiple and diverse end user devices such as Desktop PCs, PDAs, and cell phones, by a single source and suitable transformations rather than allowing for very specific animation features and the like.

For the assembly, the personalization engine uses the parameters for document structure, the layout and style parameters, and other rules and constraints that describe the structure of the personalized multimedia presentation to determine among others the temporal course and spatial layout of the presentation. The center of Figure 5.2 sketches this temporal and spatial arrangement of selected media elements over time in a spatial layout following the document structure and other preferences. The tasks for selecting and assembling relevant media elements are not necessarily conducted in a sequential manner as the Figure 5.2 shows. However, typically the selection of media elements and their assembly into personalized multimedia content is conducted in many small steps.

Transform and Present Only in the transformation phase—which we call the last mile—the multimedia content in the abstract representation model is transformed to the syntax and features of the concrete presentation formats [SB05d]. The concrete output format of the multimedia presentation is selected according to the

user's preferences and the capabilities of the end device, i. e., the available multimedia players and the multimedia presentation formats they support. Finally, the just generated personalized multimedia presentation is delivered to the (mobile) end device for the actual rendering and consumption by the users.

The process chain in Figure 5.2 depicts the general steps and phases for creating personalized multimedia content. Although it draws a very general picture of the authoring process for personalized multimedia content, it illustrates the support needed to develop applications generating personalized multimedia content for different (mobile) end devices. We find appreciated solutions and technologies in the field of personalized multimedia content generation. However, a solution approach for an efficient and simple creation of the actual personalization engine as shown in Figure 5.2 is still lacking [Bol03a].

It is important to note that the transformation step is not mandatory and thus does not exist in some systems. This is the case, when applications do not need to deliver multimedia content in more than one presentation format. In such systems, the multimedia content is already composed in the final multimedia presentation format. For example, the HotStreams system [HVL01] presented in Section 3.2.4 employs the SMIL format for internally composing the provided multimedia content.

5.2 Requirements to the Envisioned MM4U Approach

The overall goal of the MM4U approach is to improve and to simplify the development process of personalized multimedia applications by providing a comprehensive support for creating personalized multimedia content. By this, the development of personalized multimedia applications shall be made more efficient and economic. From the application scenarios presented in Section 2, we conclude that such a comprehensive support for an economic development of personalized multimedia applications needs to provide for the most different personalization aspects and features. This means, that the content not only needs to be adapted to the technical characteristics of the used end device as well as the location and environment. However, it also needs to be adapted to the user's preferences, interests, and needs. As a consequence, the proposed MM4U approach needs to be applicable for different personalized multimedia applications and thus needs to be independent of the concrete applications' domain, e. g., mobile tourism, sports news, and e-learning. We further aim at integrating the different approaches for multimedia personalization identified in Section 3.3, which are personalization by programming, templates and selection instructions, adaptive document models, transformations, constraints, rules, plans, and knowledge bases, as well as calculi and algebras. This means that we not only aim to be independent of the application domain but also allow to integrate and embrace different approaches for the actual personalization with our work. From the requirement of providing application-independent support for the authoring of personalized multimedia content and our aim to integrate the different approaches for multimedia content personalization, we conclude that for the MM4U approach the development of a software framework should be pursued. As the employed functionality shall be integrable on different levels of the MM4U framework and the

functionality provided shall be exchangeable, a component framework seems to be best suitable.

In contrast to the development of traditional software, it is not possible to provide a detailed requirements specification (see, e. g., [Som04, ch. 6]) to a software framework in advance. This is due to the typical characteristic with developing software frameworks that the abstract framework functionality emerges during the development process (see Section 4.2.1). Thus, only a high-level description of the requirements to the envisioned MM4U approach can be provided.

The requirements to the envisioned MM4U approach are derived from the application scenarios presented in Section 2. They are also derived from the previous analysis of related work, i. e., the systems for personalized multimedia content authoring presented in Section 3.2, and the systematic categorization of the related work in Section 3.3, which led to the general process chain for personalized multimedia content presented above. Following the terminology of Sommerville [Som04], the requirements to the MM4U approach distinguish functional as well as non-functional aspects. These are presented in the following Sections 5.2.1 and 5.2.2. In addition, in Section 5.2.3 the product requirements and organizational requirements to the MM4U approach are described.

5.2.1 Functional Requirements

The functional requirements to the MM4U approach are derived from and concerned with the different tasks and phases involved with the general process chain for authoring personalized multimedia applications presented in Section 5.1. These requirements are described in the following along the phases of the general process chain, beginning with the access to user profile information and media data, the composition of multimedia content, up to the transformation and delivery of the final multimedia presentations.

Access to User Profile Information and Media Data To provide authoring of personalized multimedia content, the MM4U approach needs access to user profile information as well as media elements with their associated meta data (see Section 3.1.4). Only by integrating user profile information, i. e., information and assumptions about, e. g., the user's interests, preferences, needs, as well as environment, situation, and used end device, a multimedia application is able to provide for personalized content. This user profile information and assumptions can be either explicitly collected or implicitly generated (see Section 3.1.4). To actually author the desired personalized multimedia content, the media elements must carry a sufficient amount of meta data. This meta data allows the personalized application to select the most appropriate media elements for a specific user from the media pool storages. It provides the basis for composing and assembling the selected media elements into a coherent multimedia presentation that best meets the user profile information and context information.

Basic Multimedia Composition Support and Extensibility The MM4U approach must provide for the composition of media elements into coherent multimedia presentations. Here, an abstract multimedia content representation model shall be

employed (see Section 5.1), in order to provide a basis for the generation of the multimedia content into different presentation formats and for different end devices. The MM4U approach shall support a set of basic functionality for multimedia composition and multimedia personalization. In addition, it shall be extensible in regard of more complex and application-specific multimedia composition and multimedia personalization functionality. For creating such complex or application-specific multimedia personalization functionality, the multimedia composition support of the MM4U approach shall provide for exploiting the different personalization approaches as identified in Section 3.3, as they have their individual advantages and disadvantages.

Multi-channel Provision of Multimedia Content From the related work, we derive that more and more applications share the problem that they face a big variety of heterogeneous devices, platforms, multimedia presentation formats, and players for multimedia [SB05d]. Consequently, the MM4U approach shall provide support for different (mobile) end devices and presentation formats. Although, we see a predominance of some platforms and multimedia formats on the market, it is a clear observation that not one single platform or multimedia format will prevail but rather a set of device settings and configurations will peacefully coexist [SB05d]. As a consequence, the MM4U approach shall provide support for transforming the multimedia content represented in the internal content model into different (mobile) presentation formats in order to provide a multi-channel generation of multimedia presentations to different (mobile) end devices [BH05, Top02].

5.2.2 Non-functional Requirements

Besides the application independence, we derived in regard of non-functional requirements three further aspects that shall be considered by the MM4U approach. These are concerned with the dynamic authoring of the personalized multimedia content, the issue of presentation independence, as well as the application of standardized presentation formats and existing multimedia player software. The non-functional requirements are presented in the following.

Dynamic Authoring of Personalized Multimedia Content The MM4U approach shall provide on-the-fly authoring of personalized multimedia content. This requirement is derived from the related work (see Section 3.2) and the fact that a manual authoring of personalized multimedia content is not feasible due to the huge number of different variation possibilities involved (see Section 1.1). By this, concrete personalized multimedia applications will be able to reflect the latest user profile and context information immediately and thus present the most appropriate multimedia content to the end user. With personalized multimedia applications, i. e., when collecting and exploiting user profile information, it is important that the latest profile information is reflected immediately by the application's personalized content and user interface. Otherwise a user might not be satisfied by the application's personalization results.

Presentation Independence Another very central non-functional requirement is that the MM4U approach shall provide for presentation independence in regard of, e. g., the characteristics of the end devices, their network connection, and the different multimedia presentation formats that are available. With presentation independence is meant that the MM4U approach shall support the dynamic authoring of equivalent multimedia content for the different users and output channels, i. e., presentation formats and end devices, and their individual characteristics.

Employ Standardized Presentation Formats and Existing Player Software To provide for an efficient and economic development of personalized multimedia applications, the personalized multimedia content shall be generated by the multi-channel approach (see Section 5.2.1) in standardized multimedia presentation formats. Under standardized multimedia presentation formats, we summarize formats that are defined by the W3C such as SMIL and SVG but also the ISO standard LASeR. However, also the industry-standard Flash shall be supported. In addition, supporting profiles of the standardized formats such as the Basic Language Profile of SMIL and Mobile SVG targeted at mobile devices such as PDAs and cell phones will enable the MM4U approach to reach the goal of supporting different end devices. By providing support for standardized formats, personalized multimedia applications can use and rely on existing multimedia presentation software. Thus, we avoid to develop just another new multimedia presentation format. In addition, we are also alleviated from the burden of developing yet another multimedia player.

5.2.3 Product Requirements and Organization Requirements

In this section, the product requirements and organization requirements to the MM4U approach are considered. The product requirements [Som04] include the requirements in regard of performance and portability of the MM4U approach. The organizational requirements [Som04] comprises the design and implementation requirements, process improvement requirements, as well as external requirements.

Performance and Portability Requirements Concerning performance requirements, the generation of multimedia content shall be fast (high performance) in order to employ the MM4U approach on web servers with high load and to execute the MM4U approach on devices with limited resources, such as PDAs and cell phones. In regard of portability requirements, the MM4U approach shall be portable to different stationary platforms and mobile devices (with some restrictions on the functionality on the mobile device). It shall be executable on different Java-enabled operating systems such as UNIX, Macintosh OS X, Microsoft Windows, as well as Microsoft Windows CE and Symbian OS.

Design and Implementation Requirements The design and implementation of the MM4U approach shall follow modern software engineering concepts. This includes using component technology and applying design patterns. In addition, a modern and sophisticated integrated development environment (IDE) shall be used. The implementation of the MM4U approach shall be conducted by using Java [Sun06].

Java is a widely used programming language and software platform which is available for almost every operating system and hardware platform today. The implementation shall follow common Java style guides in regard of coding conventions [VAB+00, Sun99] and source code documentation [Sun04].

Process Improvement Requirements The envisioned MM4U approach shall provide support for a more efficient and economic development of personalized multimedia applications in order to improve the development process of such applications. Consequently, this shall lead to reduced development time and development costs of such applications. In addition, with pursuing the development of a software framework, the MM4U approach shall provide for a better maintenance and extensibility of personalized multimedia applications.

External Requirements In regard of external requirements, the interoperability of the MM4U approach with systems from other organizations is considered [Som04]. From the related work, we see that the most different data models, formats, and approaches are used for the different tasks of authoring personalized multimedia content. In order to provide for a big variability, the MM4U approach shall provide for integrating different existing and future solution approaches and systems on different levels. For example, different storage solutions for user profile information and media data shall be integrated. In addition, it shall also support the integration of other existing systems and approaches for multimedia composition and personalization (see basic multimedia composition functionality and extensibility in Section 5.2.1) as well as existing multimedia presentation software.

The MM4U approach shall provide for deploying the tasks and phases of the general process chain for authoring personalized multimedia content on different (possibly remotely connected) computers. This enables the MM4U approach to serve for, e. g., digital rights management of media data and managing intellectual property rights in the case of very specific multimedia composition and personalization functionality by an external organization. The provided functionality shall also be exchangeable. This means that the implementation of the tasks and phases of the general process chain shall be substitutable by other approaches and implementations.

As the existing approaches and solutions shall be integrable at different levels of the MM4U framework, the provided functionality shall be deployable on different computers, and the functionality shall be exchangeable, a component-based approach seems to be best suitable for the MM4U framework. Consequently, the tasks of the general process chain for creating personalized multimedia content shall be encapsulated by appropriate software components. Each component provides distinctive support for one of the single tasks in the general process chain for creating personalized multimedia content.

5.2.4 Summary

In this section, we identified the requirements to the envisioned MM4U approach. Having analyzed the related systems in Section 3.2, we can say that so far there is no existing (off-the-shell) solution to buy nor research approach available that serves

all the requirements described above. In particular, there is no all-embracing research solution like the proposed MM4U approach that integrates all the different approaches for multimedia personalization presented in Section 3.3. On the basis of the general process chain for multimedia content personalization and the requirements identified, we present in the following section the architectural design of the MM4U approach.

5.3 The Layered Architecture of the MM4U Framework

Having introduced the steps and phases involved with the general process chain of multimedia content personalization and the identified requirements to the integrated MM4U approach, we now determine the architecture of the MM4U approach and describe its functionality. As stated in the introduction of Section 5.2, we aim at pursuing with the MM4U approach the development of a software framework, the MM4U framework, to provide a dynamic authoring of personalized multimedia content. The general idea of the MM4U software framework is illustrated in Figure 5.3. A personalized multimedia application uses the functionality of the framework to create personalized multimedia content and extends this functionality by whatever application dependent functionality is needed.

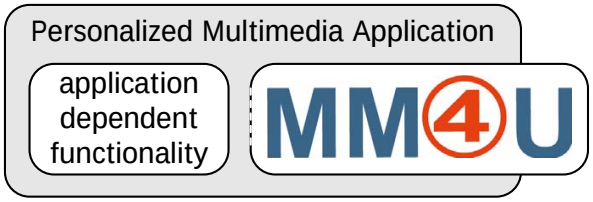


Figure 5.3: The general idea of the MM4U framework

The overall goal of the MM4U approach is to simplify and to improve the development process of personalized multimedia applications. Thus, the MM4U approach has to provide application developers with an extensive support for the different tasks and phases involved with the general multimedia content personalization process as presented in Section 5.1. For supporting the different tasks and phases, it seems to be best suited to employ a layered architecture for the MM4U framework. This layered architecture evolves from the general process chain for creating personalized multimedia content and is illustrated in Figure 5.4. Each layer provides modular support for the different tasks of the multimedia personalization process. The access to user profile information and media data are realized by the layers (1) and (2), followed by the two layers (3) and (4) for composing the multimedia content in the internal representation model and transforming it into the concrete presentation formats. The top layer (5) realizes the rendering and display of the multimedia presentation on the end device. To be most flexible in regard of the different requirements concrete personalized multimedia applications can have, the

framework's layer allow for extending the functionality of the MM4U framework by embedding additional functionality as indicated by the empty boxes with dots.

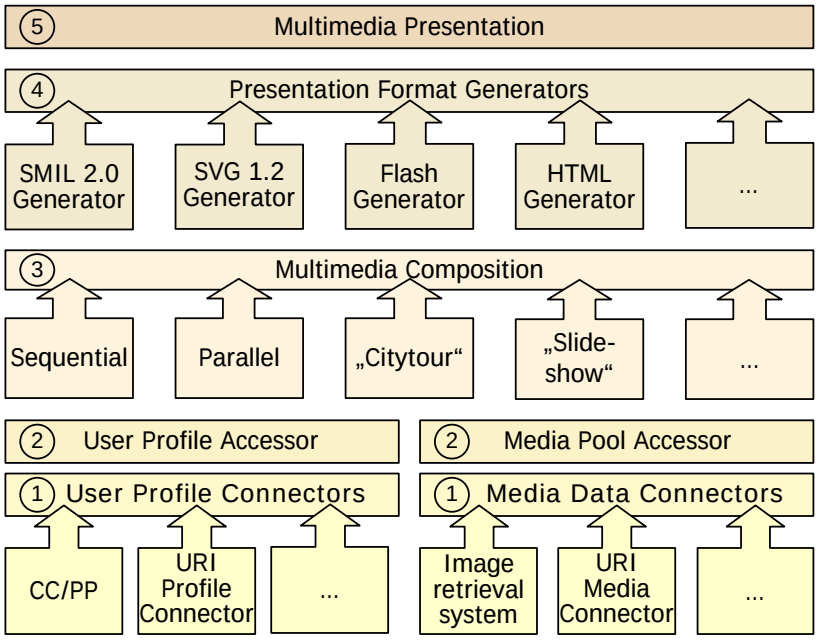


Figure 5.4: Layered architecture of the MM4U framework

In the following, the features, i. e., the functionality and objectives of the framework are described along its layers. We start from the bottom of the architecture and end with the top layer:

- The User Profile Connectors and Media Data Connectors layer (1) bring the user profile data and media data with their associated meta data in the framework. They integrate existing systems from research and industry for user profile stores, media storage, and media retrieval solutions. As there are many different systems and formats available for user profile information, the User Profile Connectors layer abstracts from the actual access to user profile information and provides a unified interface to the profile information. With this layer, the different formats and structures of user profile models can be made accessible via a unified interface. For example, a flexible URIUserProfileConnector we developed for our demonstrator applications gains access to user profile information over the Internet using the ftp protocol and on the local hard drive using the file access. However, as shown in Figure 5.4 also a User Profile Connector for the access to, e. g., a Composite Capability/Preference Profile (CC/PP) [KRW+04] server could be plugged into the framework.
  On the same level as the User Profile Connectors, the Media Data Connectors abstract from the access to media elements and their associated meta data. Here, the Media Data Connectors layer provides access to different media storage and retrieval solutions that are available today via a unified interface. For

example, a URIMediaElementsConnector we developed for our demonstrator applications provides a flexible access to media objects and their associated meta data from the Internet via the http and ftp protocol. By analogy with the access to user profile information, another Media Data Connector plugged into the framework could provide access to other media and meta data sources, e. g., an image retrieval system like IBM's QBIC [IBM04b]. The Media Data Connectors layer supports the query of media elements by the client application (client-pull) as well as the automatic notification of a personalized multimedia application when a new media object arises in the media database (server-push).

- The User Profile Accessor and the Media Pool Accessor layer (2) provide the internal, abstract data models of the user profile information and media data information with the associated meta data within the framework. Via this layer, the user profile information and media data needed for the desired content personalization is accessible and processable for the application.

The Connectors and Accessors layers are designed such that they are not reinventing existing systems for user modeling or multimedia content management. They rather provide a seamless integration of the systems by distinct interfaces and comprehensive data models. In addition, when a personalized multimedia application uses more than one user profile database or media database, the Accessor layer encapsulates the resources so that the access to them is hidden from the client application.

- The Multimedia Composition layer (3) resides on top of the User Profile Accessor and Media Pool Accessor layer. It comprises generic composition operators in compliance with the composition capabilities of multimedia composition models like SMIL [ABC+01b], Madeus [JLR+98], and ZYX [BK01, Bol01], which provide for complex and flexible multimedia composition functionality. With these composition operators and the information this layer gains from the User Profile Accessor and the Media Pool Accessor layer, an application can carry out the actual composition of the personalized multimedia content. As introduced in Section 5.1, the multimedia content is composed and assembled in an internal multimedia representation model abstracting from the different, concrete presentation formats we find today. The Multimedia Composition layer is developed such that it enables to develop additional, possibly more complex or application-specific composition operators. These newly developed operators can be seamlessly "plugged-in" into the framework's composition layer and can be used by any personalized multimedia application.
- The Presentation Format Generators layer (4) works on the internal representation of the multimedia content provided by the Multimedia Composition layer. It transforms this content into the different (standardized) presentation formats like SVG, SMIL, and Flash, which can be displayed by the corresponding multimedia player software on the (mobile) client device. In contrast to the Multimedia Composition layer, the Presentation Format Generators layer is completely independent of the concrete application domain and only relies on the targeted output formats.

In the Presentation Format Generators layer, personalization takes place such that the output format of the multimedia presentation is selected according to the user's preferences and the capabilities of the end device, i. e., the available multimedia players and the multimedia presentation formats they support.

- The Multimedia Presentation layer (5) on top of the framework realizes the actual rendering and playback of the personalized multimedia presentation. The goal here is to integrate existing presentation software of the common multimedia presentation formats the underlying Presentation Format Generator layer produces. Here, the application developers benefit from the fact that only players for standardized multimedia formats need to be installed on the user's end device and that they must not invest any time and resource in developing an own render and display engine for their personalized multimedia application.

Following the requirement of integrating different existing and future solution approaches and systems in Section 5.2.3, the layered architecture of the MM4U framework allows being easily adapted to the particular requirements that can occur in the development of personalized multimedia applications. For example, special User Profile Connectors as well as Media Data Connectors can be embedded into the MM4U framework to integrate the most diverse and individual solutions for storage, retrieval, and gathering for user profile information and media data. With the ability to extend the Multimedia Composition layer by complex and application-specific composition operators, arbitrary personalization functionality can be added to the framework. The Presentation Format Generators layer allows integrating any output format into the framework to support the most different multimedia players that are available for the different end devices.

The framework does not re-invent multimedia content adaptation and personalization but is targeted at incorporating existing research in the field and also allows to be extended by domain and application-specific solutions. The MM4U framework solves typical, often reoccurring tasks of the multimedia personalization process on an abstract level. By this, it provides application developers with a substantial support for the single tasks of the general multimedia personalization process. Although, we defined a coarse-grained architecture of the MM4U framework with the layers here, it is at this point not clear how many components this framework will finally have. In addition, it is not known how these framework components relate to the presented layers. These software engineering aspects and questions of selecting and defining appropriate MM4U framework components are discussed when the actual development and implementation of the framework is presented in Section 8.

5.4 Summary

In this section, the general multimedia personalization process has been introduced. Based on this process, the general requirements to the MM4U approach have been derived. This process and the derived requirements lead to the presented layered architecture of the MM4U framework. The functionality of each of these layers has been briefly described. The conceptual design and underlying data structures to the framework's layers are presented in the following section.

6 Design of the MM4U Framework

Having introduced an overview of the layered architecture of the MM4U framework and its functionality, we present in this section the conceptual basis and data structures forming the layers in full detail. In the following Section 6.1, the modeling of media data and meta data is considered. In Section 6.2, the modeling of user profile information and context information is described. In Section 6.3, the multimedia composition functionality of the MM4U framework is presented. Finally, the transformation to the final presentation formats is considered in Section 6.4. For realizing the top layer of the framework, i. e., for presenting the personalized multimedia content, existing multimedia players are exploited. For each layer of the MM4U framework, we discuss the modeling options that stem from the application scenarios and the existing approaches in the field, before we present the data models for the framework layers.

6.1 Modeling of Media Data and Meta Data

For modeling media data, we consider today's media types and media formats used in multimedia documents, such as the formats PNG, JPG, and GIF for the image type, MP3 and WAV as formats of audio type, and AVI, MOV, and MPEG of video type. We identified a set of four media types that can be used as input to the multimedia personalization framework. Following the distinction between discrete and continuous media elements in Section 3.1.1, these four media types are the two discrete media types image and text and the two continuous media types video and audio. As defined, an instance of such a medium type is called medium element.

- Image type: The image type is a two-dimensional array of pixels [GT95] determining the image elements' content as well as spatial extension in regard of their width and height. Examples for image elements include photos and hand-drawn items [HMHGK01]. These can be, e. g., taken by a digital camera or created with an image editing application. An image element is typically included into a multimedia document by providing a reference to the image medium in form of a Uniform Resource Locator (URL) [Net94a]. However, some multimedia document models do also allow for embedding the image element's media data directly into the multimedia presentation, e. g., SVG and Flash.
  The image type is also called graphics [BJSF94] or is distinguished between image elements and graphic elements. For example Gibbs and Tsichritzis [GT95] differentiate between the both by introducing images as defined above

and consider graphics as visual entities with some rendering options defined on it and which are typically used for displaying three-dimensional objects. Rakow et al. [RNL95] consider images also as defined above, however define graphics as consisting of lines, regions, and texts elements.

- Text type: The text type is a representation of information using an alphabetical symbol system [HMHGK01]. As an image element, also a text element has a two-dimensional spatial extension. However, in contrast to image elements, until today there is no commonly used format for describing text elements. This holds especially for formatted text that contains information about the text's style, font, character size, text color, and background color. Here, today's multimedia presentation formats pursue different solutions for modeling formatted text. For example, HTML-tags are used with the Flash format, RealText [Rea01] is employed for SMIL documents in conjunction with the RealPlayer [Rea06], and with SVG other format specific tags are used. A text element is usually directly embedded into today's multimedia presentation formats.
- Audio type: The purpose of the medium type audio is to carry audible information. It includes spoken words (speech) as well as generated tones forming music or other sound effects (see *sound* in [HMHGK01]). An audio element has a temporal extension defined by its playback duration. It has no visual part. Like image elements, also audio elements are embedded into today's presentation formats by providing a URL. However, Flash also allows embedding the audio elements' media data directly into the presentation.
  We do not distinguish between an audio type and music type, like for example done in [GT95] for distinguishing digital audio streams and MIDI-based music. In addition, we also do not distinguish between the sound type and speech type [BJSF94] or the audio type and speed type [RNL95].
- Video type: The video type captures motion pictures or animations (see *motion* in [HMHGK01]). As the visual media types image and text, also the video type has a spatial extension in regard of the video elements' width and height. In addition, it has also a temporal extension defined by its playback duration. By this, it combines the properties of the media types image and audio [RNL95]. A video element is typically included into today's multimedia presentation formats by providing a URL.
  Animations are associated to the video type, as their basic attributes are identical. Both have a width, height, and duration. An example for animations are Flash files, which are widely used in the Internet today.

With modeling of meta data for describing the media elements and by providing access to this meta data, the MM4U framework enables concrete personalized multimedia applications to select the most appropriate media elements for a specific user from the media pool storages (see the requirement for accessing user profile information and media meta data in Section 5.2.1). In regard of meta data modeling for media data, we studied different approaches of modeling meta data as well as approaches for meta data standards for multimedia. The considered approaches are, e. g., Dublin Core [Dub02] for (X)HTML [XHT05, RLHJ99] and Dublin Core Extensions for Multimedia Objects [Hun99], Resource Description Framework [BM03] used by the meta-information module

of SMIL [ABC+01b], and the MPEG-7 Multimedia content description standard [ISO99, ISO01f, ISO01b, ISO01c, ISO01d, ISO01e].

For the media types defined above, we determined a set of meta data attributes that are minimally required to apply the media elements in the multimedia composition task. To support the management of other meta data, e. g., Dublin Core, EXIF header information [Tec02] extracted from JPGs, and any other additional meta data provided by a media source, a generic concept of media elements meta data is defined. As shown in Figure 6.1, arbitrary *key-value*-pairs of meta data can be associated to a medium type. This allows for integrating the most specific and application-dependent meta data into the framework. These meta data key-value-pairs can also be recursively defined.

The UML diagram in Figure 6.1 depicts the four media types and their associated meta data attributes. The four media types are derived from the generic medium type Medium. This generic medium type defines an attribute for specifying the media data source in form of a Uniform Resource Identifier (URI) [Net94b]. A URI is a more generalized version of a URL. According to the definition above, the media types derived from the abstract Medium have additional type-specific meta data attributes such as width, height, or duration.

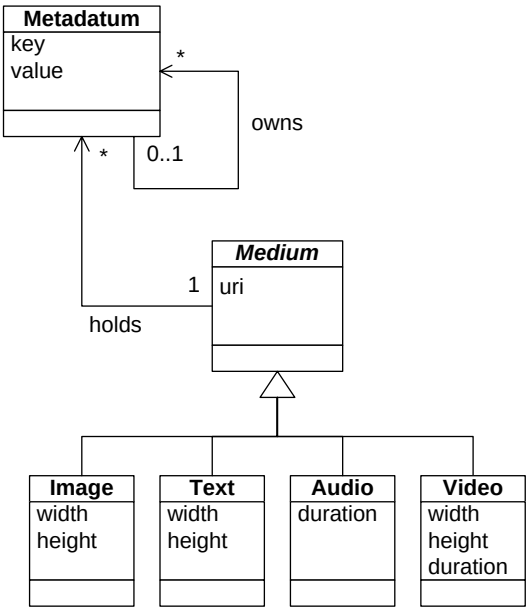


Figure 6.1: UML diagram of the media types defined in the MM4U framework

6.2 Modeling of User Profile Information and Context Information

Taking information about the user's personal profile and context into account is a necessary prerequisite for creating and providing personalized applications and services. However, in literature there are many different definitions of what user profile information and context information is. Pieces of information that are considered here are among many others information about the user's knowledge, preferences, abilities and disabilities, as well as information about the user's current situation, environment, and surrounding, including the current location as well as the characteristics of the used end device.

In the following Section 6.2.1, we first clarify the meaning of user profile information and context information and provide a definition of a unified user model that includes both. Following this definition of user model, we present in Section 6.2.2 our comprehensive abstract user model. The definition of this abstract user model bases on an extensive study of related approaches and standards in the area of user modeling and context modeling. This flexible and hierarchical user model is independent of a concrete application domain. Due to its flexibility and extensibility, the user model can be applied for arbitrary personalized systems. The abstract user model can be used as basis to develop application-specific user models by extension and refinement.

6.2.1 Unified Approach to User Modeling

For determining the notion of user profile information and context information, we first analyze what user profile information is and present the concept of a user model. Then, we analyze context information and introduce the concept of context and context model, respectively. Finally, we present our definition of a unified approach for user modeling, which includes both user profile information and context information.

User Model For modeling user profile information and context information appropriate data structures need to be applied. For modeling user profile information the corresponding data structure is called user model. For the term user model, an abstract definition is given by Wahlster and Kobsa [WK89, Kob96] for the area of natural-language dialog systems, where the research on user modeling originated [Kob93b]. They define a user model as a distinct part of an application that contains explicitly modeled information and assumptions on all aspects of the user that may be relevant in regard of the dialog behavior of the user with the application (see also [FKS97a, Kob95]). This means that with a user model information about the user is represented within a computational environment and thus is made machine processable (cf. [Fis01]). Consequently, employing a user model enables applications to differentiate and distinguish among different users [BM02, All90].

The focus of the research done in the area of *classic* user modeling lies in the modeling of mental and cognitive characteristics of a user (see, e. g., [BM02, KKP01, FKS97b, Kob96]). The goal is to gain information about the users and their behavior, interests, preferences, and knowledge by monitoring and analyzing the users'

interaction with the application [BC01]. Consequently, the core aspects modeled are the users' (existing and missing) knowledge, goals, plans, interests, preferences, (physical, sensorial, and cognitive) abilities, misconceptions, and other individual features [BM02, Kob95, Bru94, Kob93b, Kob93a].

Which pieces of information and assumptions about the user are actually stored and processed in a user model of a concrete application cannot be generally defined as it depends on the actual application's domain. Consequently, an enumeration of distinct user modeling aspects would not be able to cover all aspects a concrete personalized application could need to model about a user and by this it cannot be complete.

Context With the emergence of small, sophisticated mobile computers in the nineties, such as PDAs and enhanced cell phones (e. g., smart phones), also the aspect of modeling contextual information emerged [BM02]. The single pieces of contextual information are usually subsumed under the notion of context. Analogously to the user model, we introduce here the concept of a context model [SBG99] as an appropriate data structure for modeling context information. For the term of context or context model, we find many different definitions in literature. A first approach could be to define context by an enumeration of its single modeling elements. However, as with the user model, this is quite difficult since many aspects can be understood and subsumed by context. An abstract definition of context is provided, e. g., by Dey and Abowd [DA99], saying that it comprises any information that can be used to characterize the situation of an entity. An entity constitutes a person, place, or object which is relevant to the interaction between a user and an application, including the user and applications themselves. Schmidt et al. [SBG99] introduce the concept of context in a similar way as a model describing a situation and the environment a device or user is in. This context model has for each context a set of relevant features. The presented definitions consider context from different point of views, including the user, the application, and the environment. The latter includes objects, e. g., the end device, and/or persons. Schilit et al. [SAW94] provide a similar definition of context. However, it is clearly motivated by the user's current environment and situation only:

> "Three important aspects of context are: where you are, who you are with, and what resources are nearby. Context encompasses more than just the user's location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth, and even the social situation; e. g., whether you are with your manager or with a co-worker." [SAW94]

In this work, we consider context from the user's point of view, since the user is the starting point and the trigger for modeling context information. Consequently, for the definition of context, we follow Schilit et al. regarding the users context as information about their environment and situation. This includes objects as well as persons that are with you or nearby.

Unified User Modeling Approach From the analysis and discussion above, we draw the conclusion that user profile information is typically information that "comes

from and is about the user". In contrast, context information are typically information that are "gathered from the user's environment and situation". However, the distinction is not always clear and both user profile information and context information run into each other. For example, gathering and analyzing the interaction history with the application can be considered as information that is gathered by the application within the user's computational environment but as well allows to draw some conclusions about the user's behavior and knowledge.

Context information has a strong relation to the individual user of the application, e. g., the location and the used end device. Therefore, we define context information to be a specific part of the user profile information. Consequently, we define like Fink, Kobsa, and Schreck [FKS97a] the context model to be a sub-part of the user model. We support this decision by the fact that the reason for collecting context information is triggered by the goal of providing information and services that best meet the user's preferences and needs. Consequently, we pursue a unified user model approach that embraces and unifies both classic user profile information and context information. With it, we are able to simultaneously consider both, the aspects we have learned to deal with in the past decades to model information and assumptions about the cognitive aspects of the user [Jam01] and the different environments and situations a mobile user can be in today. Following, we present our definition of a user model.

A user model is described by an abstract data structure. This data structure defines what kind of information and assumptions about a user and his or her environment and situation is modeled and processed by a software system. The user model's data structure includes the aspects of both user modeling and context modeling, i. e., user profile information and context information.

On the conceptual level of managing user profile information and context information, i. e., for storing, processing, and retrieving such information, we distinguish between user models and their profiles. The user model defines an abstract data structure, whereas a user profile contains the actual information about the individual user of a concrete personalized application. Hence, user profiles can be seen as "instances" of a user model, filled in with the concrete values of an individual user.

The abstract user model can be considered as a machine-readable and machine-processable representation of the user. It captures all information that are necessary to personalize the provided services and content to the individual user needs. The extent to which personalized applications are able to individually create and deliver their content and services to the users depends on the information captured by the user model and how this information is utilized [Ped00].

6.2.2 Definition of an Abstract User Model

When designing user models, the result is often an application-dependent solution that is only applicable for a specific application scenario. However, one requirement to the MM4U framework stated in Section 5.2 is to provide support for the most different personalization aspects and features and not rely on one single user model only. We conclude that the MM4U approach must be applicable for the most different multimedia applications and thus be independent of the concrete applications' domain. This is reflected by the MM4U framework with the ability to provide access to specific solutions by developing appropriate user profile connectors and integrate

them in the framework's connector layer. However, with the user profile accessor layer, we aim at providing an internal abstract data model for the accessed user profile information, i. e., we target at providing an abstract user model. Abstract means that the user model is not designed in regard of the requirements of a specific application area but is applicable for applications from different domains.

To serve different application areas and by the fact that an enumeration of the most different aspects of user profile information and context information (as argued in Section 6.2.1) cannot be complete, the abstract user model must be most flexible. This means, that the abstract user model must provide the ability for arbitrary extensions and refinements in regard of application-specific user modeling and context modeling aspects that are required by a concrete application. Although a complete list of modeling aspects for the most different domains and applications cannot be provided, the key aspects in regard of possibly relevant user profile information shall be provided by such an abstract user model.

The data structure used for the abstract user model is a tree structure, where the single pieces of information about the users are hierarchically ordered. The nodes in this tree structure determine the kind and type of information the user model stores about the users. This are, e. g., information about the user's preferences, interests, current location and environment, demographical data, and used end device. Subtrees in this data structure constitute refinements of the parent nodes. For example, the demographical data can be refined into pieces of information such as name, address, gender, date of birth, place of birth, family status, and others. Consequently, the edges of this tree structure denote an *is-part-of*-relationship between the child nodes and their parent node. As the abstract user model is organized in a hierarchical tree-like data structure, it can be easily extended and refined to specific requirements of a concrete personalized multimedia application.

For depicting the abstract user model's data structure, UML class diagrams are applied. The root node of the abstract user model is a UML class that represents the individual user as depicted in Figure 6.2. This User class possesses an attribute that holds the identity of the user and has five other classes associated via UML's composition functionality. These five classes constitute the five main nodes of the user model, which are the users' demographical data, physical characteristics, mental characteristics, situation, and used end devices.

The abstract user model defines more than 200 nodes, from both for *classic* user modeling and context modeling, covering an extensive set of user profile information a concrete personalized multimedia application may need to exploit. The five main nodes and the corresponding sub-nodes of the abstract user model, i. e., the five main classes and the associated classes are presented in detail in the following Sections 6.2.2.1 to 6.2.2.5. To provide for very particular personalization aspects, the user model is extensible and refineable in regard of the characteristics and requirements of specific applications.

6.2.2.1 Demographical Data

The demographical data about the users includes information that is typically printed on business cards or is found in letter-headings. Here, vocabularies such as the Platform for Privacy Preferences (P3P) [CLM02], Customer Profile Exchange (CPExchange) [BH00], and vCard [DH98] comprise the most important pieces of in-

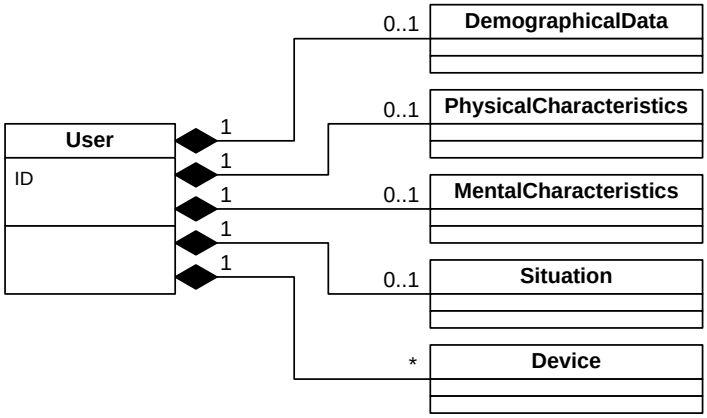


Figure 6.2: The abstract user model's root note and its five main categories

formation that needs to be considered when describing the demographical data. Figure 6.3 shows the demographical data node and its child elements, i. e., the classes associated to the DemographicalData class. These classes are described in the following.

Name Within the class Name, all information are considered that can be used to identify and to name the user. To this class, further nodes or classes are associated (not shown in Figure 6.3) that include among others the academic title, forename, surname, nickname, as well as names used to identify the user for login and registration procedures.

Contact The note Contact contains information that can be used to contact the user, e. g., by physical mail, e-mail, (mobile or Internet) phone number, or instant messaging. Consequently, the ContactData class has further classes associated, which comprise among others the address of the user, consisting of the street name, street number, zip code, and country, as well as regular phone number, cell phone number, e-mail address, but also the fax number and Internet homepage of the user.

Gender, Birthplace, Date of Birth, Nationality, and Denomination These nodes comprise information about the gender, birthplace, date of birth (which allows to determine the user's age), nationality, and denomination of the user. The corresponding classes constitute leaf nodes of the abstract user model. This kind of information is used, e. g., in the domain of tourism to adapt the style, layout, and colors used to the cultural area of the user [Zip02].

Family To the node Family, further classes are associated that carry information about the parents of the user, children, the partner, and the martial status of the user.

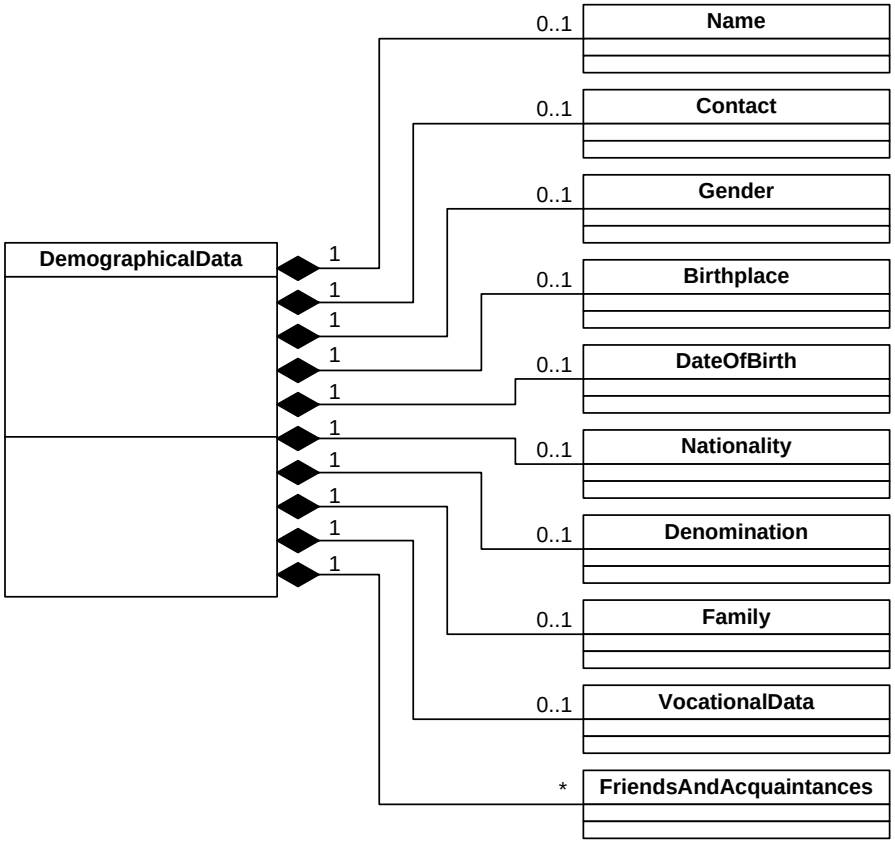


Figure 6.3: Demographical data of the user

Vocational Data The node VocationalData comprises all information that correlates with the profession of the user. This information is modeled by associated classes that include among others the occupational title, the employer, the department or division the user works with, the line of business, as well as information about the professional experience and education.

Friends and Acquaintances The node FriendsAndAcquaintances holds a list of associated nodes describing the friends and acquaintances of the user. This kind of information can be used, e. g., to notify the user when a friend or acquaintance enters or leaves the user's *context* (also refferred as buddy function).

6.2.2.2 Physical Characteristics of the User's Body

The second main node of the abstract user model is the PhysicalCharacteristics class. It comprises all information that are in relation with the physical characteristics of the user's body. As depicted in Figure 6.4, this information is divided into five parts.

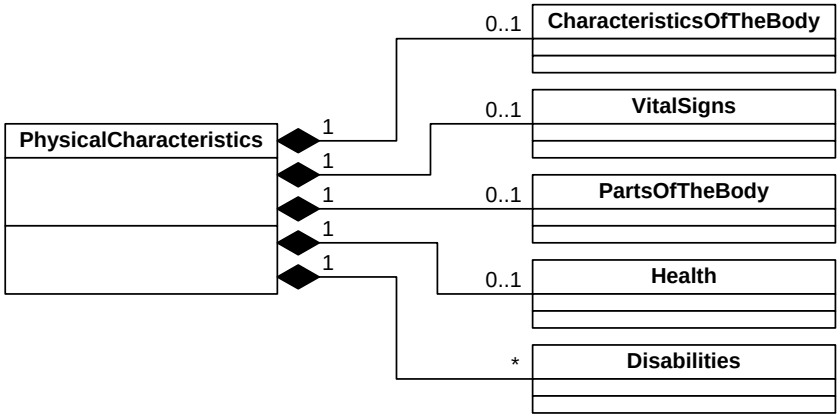


Figure 6.4: Physical characteristics of the user's body

Characteristics of the Body The general characteristics of the body embrace the outer and inner characteristics of the user's body. These are associated to the CharacteristicsOfTheBody as further nodes and comprise information such as body height, weight, hair color, eye color, dress size, size of shoe, as well as blood type.

Vital Signs Within the child nodes associated to the node VitalSigns information about the electrical skin resistance, blood pressure, breathing frequency, pulse rate, body temperature, and pupil widening are stored. Possible applications here are, e. g., automatically calling an emergency medical assistance service if specific vital signs excess or fall below of threshold values. From vital signs, also information and assumptions about the emotional state of the user can be derived [Pic98].

Parts of the Body The node PartsOfTheBody has further nodes associated that describe the current state of the singular parts of the body such as head, arms, and legs, as well as the position of the body, e. g., upright position or horizontal position. This kind of information has a short term validity, such as the current agitation of the left hand or the angle of vision of the head. A possible application scenario is that the user is offered to provide a natural language input to a system instead of using some haptic input devices, if the user has currently no hands free [Jam98].

Health This node of the user model embraces further nodes associated that carry information relating to the user's health. These are among others diseases, allergies, and vaccinations. A personalized system that compiles the daily food of the user can take food allergies of the user into account. Information about the user's health is considerably sensitive data. Consequently, a personalized application gathering and exploiting such critical information needs to take corresponding security issues into account.

Disabilities The node Disabilities comprises associated classes with information about physical disabilities of the user, e. g., amputation of extremities, color-

blindness, and visually or auditory handicaps. Exploiting this information, the user interface of the application can be adapted to the abilities and disabilities of the user [KLCR02]. A personalized application would take care of these disabilities of the user, e. g., by using large font sizes for visually impaired, switching the presentation's modality to visual for auditory handicapped, using appropriate colors for users with color-blindness, or choosing a route without stairs for a person in a wheelchair.

6.2.2.3 Mental Characteristics of the User

The mental characteristics of the user comprises information as defined from the classical user modeling research community. This includes information about the user's knowledge, goals, plans, interests, preferences, abilities, and misconceptions (see Section 6.2.1). It also comprises information about the habits, personality, emotional state, mood, and current ability to pay attention. This results in a MentalCharacteristics node as depicted by the UML diagram in Figure 6.5.

In contrast to the other core modeling nodes of the abstract user model, the mental characteristics of the user described in this section are considered as soft information. This means that this kind of information, e. g., knowledge, personality, mood, and emotions, is not directly measurable or determinable. In contrast, hard information is typically directly measurable, e. g., the body weight, height, and blood type.

Knowledge From a psychology point of view, knowledge is considered as relatively long-lasting information in the user's long-term memory. Hence, knowledge is understood as subjective, justified true belief [RN03]. The knowledge a user has is modeled by the name of the domain or the specific topic and the corresponding knowledge level [BE98] the user has in this domain or topic. These are modeled in the abstract user model as a list of nodes associated to the Knowledge class. Acquiring information about the knowledge a user has in a specific domain or topic is highly complex and thus the reliability of the results is less than certain [KKP01]. Consequently, information about the user's knowledge is often referred as assumptions an application has about the user's knowledge. Thus, in this work knowledge is considered as the information and assumptions a personalized multimedia application has about the user's beliefs about concepts, relationships between concepts and facts, and rules with regard to the considered domain [KKP01]. For example, a personalized multimedia application could have the assumption that a user's knowledge level in the domain of mathematics is beginner.

In addition to the structure of knowledge described so far, knowledge can be organized hierarchically. The knowledge about a subject can be refined by knowledge about specific topics in this subject. For example, the user mentioned above could have basic knowledge in the field of mathematics in general, however high knowledge about the theorem of Pythagoras in particular.

As knowledge is defined as the subjective (although justified) belief of the user, it is possible that the user's beliefs are wrong. Thus, the user's knowledge can include misconceptions, e. g., a user beliefs that a specific construct is correct although it is proved that it is not. For example, the user could have a misconception about the

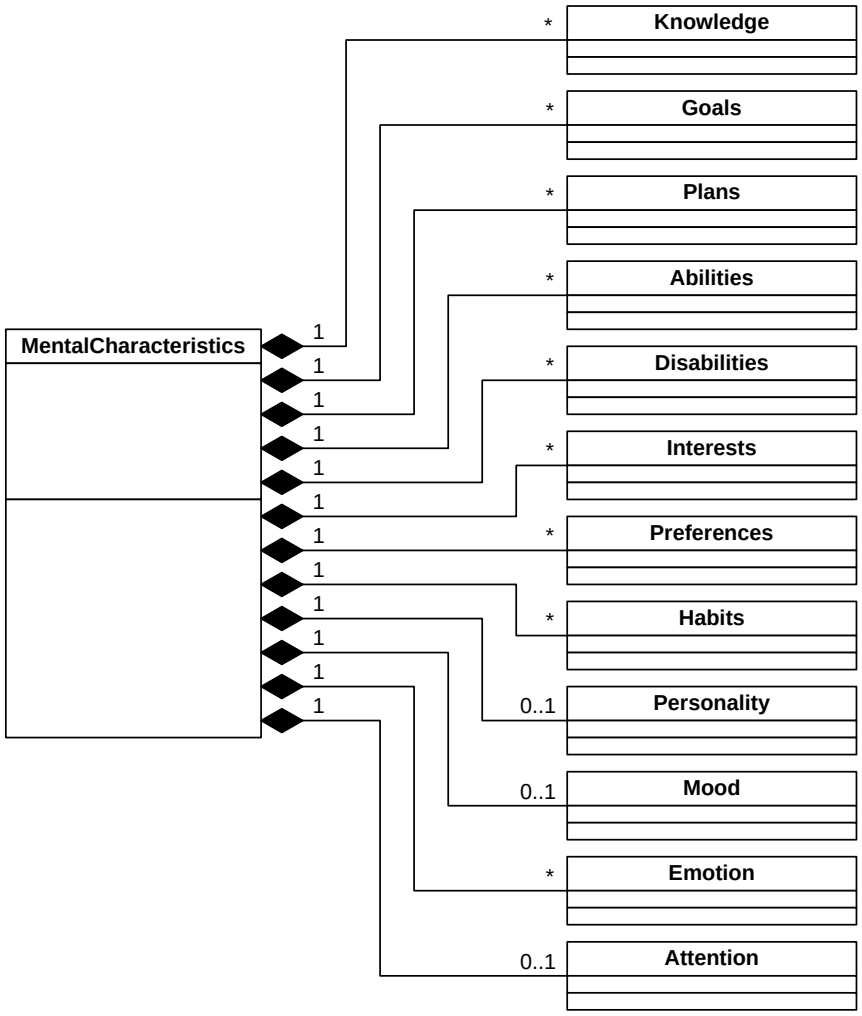


Figure 6.5: Mental characteristics of the user

theorem of Pythagoras, i. e., he or she has a notion of the theorem which contradicts to its mathematical definition.

An application scenario where information and assumptions about the user's knowledge is exploited, is, e. g., in the area of e-learning applications an automatic help system. Such a help system could answer questions on the basis of the knowledge level of the user. If the application knows that the user has none or only little prior knowledge in the domain, then the answers are presented in full detail that do not require any prior knowledge. However, if the user is an expert in the domain, the application can assume that the user has some foreknowledge and the answer can be formulated accordingly (cf. [Chi89]).

Goals The notion of goal is dependent on the area in which it is used. Opwis [Opw96] distinguishes between the two areas of motivation and problem solving: In the area of motivation, a goal is the intention for acting or behaving in a certain way to reach an anticipated notion of the impact of ones action. For the area of problem solving, a goal is typically seen as a solution state, which is predetermined by the way of posing a specific problem. It is desired to reach the solution state or also called goal state and one shall aspire for it [RN03]. In addition to an overall goal of a specific problem, a goal hierarchy can be build [RN03]. This is achieved by decomposing a goal into sub-ordinate targets or sub-goals. Single sub-goals are connected by *and*- or *or*-operations and can themselves comprise further sub-goals [RN03]. The *and*-operation means that the goal is reached, when all of its sub-goals are accomplished. For the *or*-operation the accomplishment of one sub-goal is sufficient to reach the superordinate goal.

In this work, the notion of goal is defined on the basis of both areas motivation and problem solving. A goal is defined as possessing a motivation as well as a list of sub-goals, connected by the described operators. These are modeled in further sub-nodes associated to the Goals node. As sub-goals themselves can contain other sub-ordinate targets, also a hierarchical tree structure of arbitrary depth can be defined. Examples where assumptions about the user's goals can be exploited are when users aim to find information about a specific topic or if they gain for shop some kind of product [KKP01].

Plans The notion of plan is defined as structure that is targeted for representing actions and goals. A plan is used to argue about the impact of future actions and to affect the goal-oriented actions of a user [RN03]. By this, a plan consists of several concurrent or sequential actions modeled as child nodes of the Plans class in the abstract user model. Each action may be influenced by its predecessors in the plan [Pea00]. Actions can be either reactive or deliberative. A reactive interpretation sees an action as a consequence of the user's beliefs, disposition, and environmental inputs. The deliberative interpretation sees action as an option of choice in contemplated decision making, usually involving comparison of consequences [Pea00].

Abilities and Disabilities Abilities are skills learned by education and training. They are modeled in the abstract user model as a list of associated nodes carrying the abilities' name and grade. For example, a user could have the ability *driving a car* with the grade *expert* [Ric89]. Besides the mental abilities, also the user's mental disabilities need to be modeled. Such disabilities can be, e. g., learning disabilities [SB92]. In the abstract user model, the disabilities are modeled in the same way like abilities. Thus, they are also described by their name and a grade of disability.

Interests The interests describe what the individual user likes and dislikes. Information and assumptions about interests are vital for and exploited by many personalized systems [KKP01], e. g., by the tourist guide Deep Map [FK02]. Typically, a personalized tourist guide application models assumptions about the topics a user is interested in, e. g., art, architecture, museums, or natural parks [CGL+02]. Modeling of user interests is also a central aspect for recommender systems [KKP01].

Recommender systems [SKR01, RV97] are a specific type of personalized applications targeted at making recommendations for products, goods, or services a user might be interested in, e. g., the Internet bookstore Amazon.com [Ama06].

Within the abstract user model, the user's interests are described by a name and a value in the interval $[-1, 1]$. These are modeled as a list of associated nodes of the Interests class. A value of -1 represents that the user is not interested in a particular topic, while a value of 1 shows a high interest. The value of 0 indicates that the user has no specific interest or lack of interest in that particular topic. In addition, hierarchies of interests can be build. For example, the assumption that a user has general interest in mathematics can be refined in a second-level hierarchy describing the user's particular interest for geometry.

Preferences The notion of interests and preferences is often considered to be equal. However, they are slightly different as the following example shows. A user's preference is that a user has English as preferred language for receiving multimedia content, instead of, e. g., German or Japanese. However, to the user's interest belong that a user likes art from a particular period, e. g., the antiquity. Besides this semantic difference, preferences are modeled in the same manner as the interests in our abstract user model consisting of a name and a value determining how much the user prefers or dislikes it. An example of typical assumptions about the user's preferences a personalized tourist guide application could exploit are the preferred type of hotel or restaurant [CGL+02].

Habits Typically, users show some habits, i. e., specific patterns of behavior. These can be modeled and stored within nodes associated to the abstract user model's node Habits. Example for a specific pattern of behavior is that the user usually drinks coffee for breakfast or is used to watch sports on Saturday's TV afternoon.

Personality For modeling the user's personality we base on the work done by Egges, Kshirsagar, and Magnenat-Thalmann [EKMT04, Ksh02, KMT02]. Here, personality is considered to be constant [EKMT03], i. e., it is practically not changing over time [Ksh02]. The user's personality causes deliberative reaction and affects how the user's mood changes over time [Ksh02]. Considering the modeling of the user's personality from psychology research, there are many personality models to be found [EKMT04]. These personality models consist of a set of dimensions, each describing a specific property of the personality. As the personality theories provided by the psychology research have different numbers of dimensions, it is useful to abstract from the provided personality models and assume that personality has n dimensions [EKMT04, EKMT03]. Each dimension is represented by a value in the interval $[0, 1]$. The personality dimensions are modeled as a list of nodes associated to the abstract user model's Personality node. The value of 0 denotes the absence of the specific dimension in the user's personality. On the other end of the interval, the value of 1 corresponds to a maximum presence of the specific dimension in the user's personality [EKMT04, EKMT03]. The framework proposed in [EKMT04, EKMT03, KMT02] for simulating personality, mood, and emotion of virtual humans employs the widely accepted five factors personality model [MJ92, Dig90]. For our abstract user model, we also employ this five factor model

for the modeling of the user's personality. The five personality aspects or personality dimensions defined in the five factors model are openness, conscientiousness, extraversion, agreeableness, and neuroticism.

Mood With mood, a conscious and prolonged state of mind is understood that directly controls the user's emotions [Ksh02, KMT02, Wil99]. It is clearly distinguished between mood and personality. Personality is considered as causing deliberative reactions. These reactions in turn causes the mood to change [Ksh02]. In addition, the user's mood is also affected by the momentary emotions as a cumulative effect [Ksh02, KMT02, Wil99]. Mood is considered as a mid-term characteristic of the user's mental state as it is less static than the user's personality. However, it is also less fluent than the user's emotions, which can change very quickly and thus are considered as short-term characteristics [EKMT04].

For modeling mood a one-dimensional structure can be applied, such as the user feels good or bad, or something between [Ksh02]. However, like the user's personality the mood can be multi-dimensional, like a user feeling in love and being paranoid [EKMT03]. Whether mood should be modeled one-dimensional or multi-dimensional is beyond the scope of this work (cf. [EKMT04]). However, to provide for most flexibility it is reasonable to model mood as possibly having multiple dimensions [EKMT03]. Each dimension is described by a value that is either negative or positive and lies in the interval $[-1, 1]$. These dimensions are described in the abstract user model by a list of nodes associated to the user model's Mood node. Using the one-dimensional structure for modeling mood from above, a value of -1 would represent that the user is in a bad mood, the value of 1 is interpreted as feeling good, and the value of 0 is considered that the user is in a neutral mood.

Emotions Emotions are an integral part of the human's decision making system. They are short-lived and decay quickly [Wil99]. Hence, they are associated to the user's short-term characteristics. For the modeling of the user's emotions, we base—like with the notion of personality and mood—on the work by Egges et al. [EKMT04, EKMT03]. The structure used to model the emotional state of the user is similar to the structure for modeling the personality [EKMT03]. The emotional state is considered as a set of emotions where each emotion is representing a distinct dimension in the emotional state [EKMT03]. Each emotion has a certain intensity, represented by a value in the interval $[0, 1]$. The emotions are modeled as a list of child nodes associated to the Emotions node. The size of the set of emotional dimensions describing the user's emotional state depends on the psychological theory that is used as basis for it [EKMT04]. For our abstract user model, we employ a set of 24 basic emotions as defined in [Ksh02, KMT02]. These 24 emotions are an extension of the 22 basic emotions defined in [OCC88]. They are arranged into six groups of basic emotion expressions [Ksh02], which are joy, sadness, anger, surprise, fear, and disgust.[1]

Attention For the mental characteristic Attention it is important to consider the time the users have for a specific task. The possibility to pay attention is influenced

[1] In [Ksh02] these six groups are exploited for visualizing a virtual human's mood.

by (stress) factors from the outside world (situational distractions [Jam98]) such as if the user is currently driving a car. It is also influenced by the current state of mental exhaustion, e. g., if the user is tired. The node Attention is not specified in detail in the abstract user model and it is left to the concrete personalized multimedia application to further determine it.

6.2.2.4 Situation of the User

Within the Situation node of the user model, the current situation of the user is modeled. The user's situation is strongly related to what is defined as context in Section 6.2.1. Different aspects can be identified that correlate with the situation of the user. The most common aspects are time, location, and the characteristics of the physical environment, as defined by, e. g., [DA99, CK00, GS01]. In addition to these aspects, it is also important to consider the user's social situation. This includes information about the current role and the people that are nearby or with the user [CK00]. The definition of the corresponding Situation class in our abstract user model is depicted in Figure 6.6. The different aspects of the user's situation are described in the following.

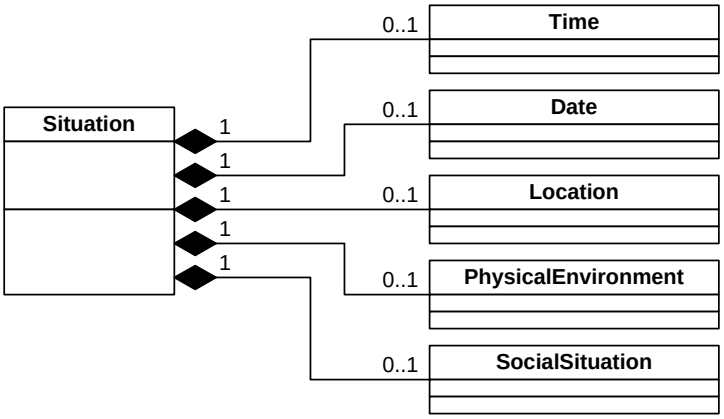


Figure 6.6: Situation of the user

Time and Date The node Time contains values describing the current time as well as the time zone the user is in. The node Date holds a value describing the date of the location of the user. Exploiting information about time and date offers, e. g., a personalized sightseeing application to add only those sights to the user's personal route that are currently open and can be visited at that specific time of day and season.

Location The location is an important part of the user's context [SHGN04]. It is typically exploited by mobile personalized applications, such as tourist guides. Typically, the users' location is understood as their current position. In this work, we extend the notion of location by comprising not only information about the user's

current position but also the current movement. Both are modeled as further classes associated to the abstract user model's Location node.

The user's position is modeled by the class Position and can be described either by providing a coordination-tuple or an hierarchical description [JS02]. Both types of information are associated to the Position node as further classes of the abstract user model. When providing a coordination-tuple, the user's position is stored in the user model as triple consisting of longitude, latitude, and altitude in a specified system of coordinates. An example for a hierarchical description of the position is the address of a building the user is currently in. This building is located in a particular city, region, and country on the earth. This allows for a hierarchical description of the position that can be provided, e. g., by employing the Getty Thesaurus of Geographic Names [The06c]. However, the hierarchical description of the position can also be refined. For example, the position can be enhanced by information about, e. g., the floor and the room of the building the user is currently in. Consequently, the position can be described by a hierarchy that reaches from the country up to the room the user is currently in. Such an hierarchical approach allows personalized applications to get different levels of details describing the user's physical location and surrounding. For example, one application might be interested in the room the user is currently in, while another application only needs to know the city the user is currently visiting. In addition to the coordination-tuple and the hierarchal description of the position, it can also be explicitly stated if the user is inside or outside a building. This information is important, as the relevant context within a building and outside a building can strongly differ. For example, the context information about the current weather is probably more important if the user is going to be outside the next couple of hours than if he or she is in a building.

The presented hierarchical description can not only be used for determining the user's current position. In can also be employed for describing the position of other people or objects that are relevant to the users. Such an approach is pursued with the augmented world model developed in the Nexus project [LBBN04, NGS+01, GLMR01].

Within the class Movement the second aspect of location, i. e., the movement of the user is modeled. The class comprises information about the direction and velocity of the movement as well as the used means of transportation (cf. [CGL+02]). This information is modeled as nodes associated to the Movement node. The direction of the movement is determined by a three-dimensional vector. The means of transportation is described by a name, e. g., bus, car, taxi, train, airplane, and the like, together with further information about the characteristics of the means of transportation. Information about the movement can be used, e. g., to adapt the output modality. For example, if the user is walking, the system could provide information via acoustic output rather than a visual presentation of a sightseeing spot. Consequently, the users do not need to stop their movement or reduce velocity in order to read the information on the screen (cf. [Jam98]).

Characteristics of the Physical Environment The node CharacteristicsOfThePhysicalEnvironment embraces different measurable physical characteristics of the environment relevant for the user. This includes weather information such as temperature, wind strength, rain, barometric pressure, air humidity, lighting conditions as well as loudness of the environment. All characteristics of the physical environment

are modeled by corresponding classes of the abstract user model, which are associated to the CharacteristicsOfThePhysicalEnvironment node. An example of exploiting information about the characteristics of the physical environment is, e. g., adapting the background lightning of the display according to the lightning conditions of the environment [Sch00]. Or if the loudness of the environment is very high, the output modality can be changed from auditive to haptic, e. g., a cell phone could switch to vibration alarm for an incoming call instead of using an acoustic signal.

Social Situation The social situation of the user is described among others by information about the user's current role and the people that are nearby or with the user [CK00, SAW94]. In regard of the role, it is important, e. g., if the user is on business or private. The current role is also strongly related to the current task. For example, if a user is on private, a role can be, e. g., going shopping in the pedestrian zone, going to a concert, or visiting a medical doctor. For the business role, it can be important to know whether you are with your boss or with a co-worker [SAW94]. These people that are with the user are modeled by appropriate nodes associated to the SocialSituation class of the abstract user model.

6.2.2.5 Device

The node Device models the technical context of the user. A device can be a stationary Desktop PC, notebook, PDA, cell phone, and the like. Since a user can simultaneously carry more than one of these devices, a set of Device nodes is provided by the abstract user model (see Figure 6.2). The attributes defined in our abstract user model to describe the end devices' characteristics mainly base on the vocabularies such as User Agent Profile (UAProf) [Ope06b], FIPA Device Ontology-Specification [FIP02], and CC/PP [KRW+04]. Figure 6.7 shows the modeling of the user's devices within our abstract user model.

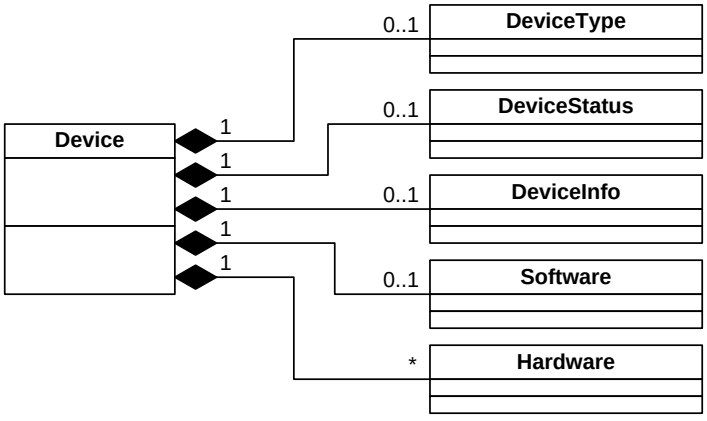


Figure 6.7: Characteristics of the devices

General Characteristics of the Device With the general characteristics of the device, we subsume the nodes describing the device's type, status, and further information. The class DeviceType denotes the kind of device. Possible values for the device type are, e. g., Desktop PC, PDA, and cell phone. The node DeviceStatus determines whether the device is currently in operation and in which state the device is currently in. The class DeviceInfo embraces further information about the device such as the name and version, the manufacturer, and the firmware version.

Hardware The description of the hardware embraces information about the single physical parts of the device. This includes information such as the available input and output devices, CPU, network interfaces, and power supplies. These characteristics of the end device's hardware are modeled by classes associated to the Hardware node. In addition, there are further hardware characteristics described such as the *pixel aspect ratio* and *screen size char* of the end device display, like to be found with CC/PP [KRW+04].

One of the central questions pursued in regard of exploiting information about the characteristics of the end device is to adapt the content of web sites and multimedia presentations to the limits of the end device's display. Another example for exploiting information about the hardware characteristics of the end device is adapting an image element to the offered resolution and color-depth of the display or using a slideshow of image elements instead of a video element, if the end device is not capable of rendering video clips.

Software The node Software comprises information such as the name and version of the operating system, the available run-time environments, and supported network protocol of the device. In addition, it also includes a list of applications that are available. This information about the end device's software is modeled as classes associated to the Software node. For the single applications installed on the device, information can be stored such as the type of the application, its name, version, support for internationalization, supported file types, and manufacturer. In addition, further possibly application-specific information can be stored for each application. For example, information stored about the installed web browser could consider characteristics such as the supported HTML version, support for frames and tables, as well as if Java or JavaScript is available.

An example for exploiting information about the available software on a device is to check which types of data can be interpreted by a specific application on the device. For example, the mobile multimedia players typically allow only a limited set of media formats for rendering image, audio, and video elements. By exploiting this information, the multimedia content can be adapted to the file formats supported by the device's software.

6.2.2.6 Summary

In the previous sections, the abstract user model and its single nodes have been introduced. The user model is defined as a combination of the core aspects of classical user modeling research as well as the aspects appeared with the availability of powerful mobile end devices, such as PDAs and cell phones, typically subsumed

under the notion of context. However, such an abstract user model can never define all (application-specific) aspects a personalized application might like or need to model about a user. To provide for such very particular aspects, the user model is extensible and refineable in regard of application-specific characteristics and requirements. By this, it provides support for the most different applications and domains of personalized multimedia applications.

6.3 Abstract Multimedia Content Representation Model

In the previous two Sections 6.1 and 6.2, the data models for media elements and meta data as well as user profile information have been presented. These models are exploited by the multimedia composition task to actually create the personalized multimedia content. From the general process chain for personalized multimedia content authoring presented in Section 5.1, we derived the central requirement that an abstract multimedia content representation model is needed to provide for a multi-channel generation of multimedia content into different presentation formats and for different platforms (see Section 5.2.1). For determining this abstract multimedia content representation model, we need to analyze today's (standard) multimedia presentation formats. Even though an earlier comparison of existing presentation formats was conducted by Boll et al. [BKW99b], this analysis is not helpful here, as it was directed towards identifying new and fancy multimedia modeling features.

At this point, we could have considered to use any of the existing sophisticated multimedia document models, which have been developed from the application point of view, as basis of our multi-channel approach to generate all the other presentation formats. But this would not ensure that the syntax and multimedia modeling characteristics of the selected model can be optimally transformed to the syntax and multimedia modeling characteristics of the different presentation formats. As the subsequent analysis shows, multimedia document models of the modern multimedia presentation formats such as SMIL and SVG unfortunately intermix the different aspects of multimedia composition, e. g., the definition of media elements and their temporal synchronization. For example, SMIL presentation specifications intermix the media elements definition and the determination of the temporal course of the presentation with the beginClip and endClip attributes. As shown in Figure 6.8, the former approaches we find in the area of multimedia document models and presentation formats are targeted at identifying sophisticated requirements and fancy features for multimedia presentation, e. g., [Pih03, BKW99b]. In contrast, with our approach we aim at analyzing the existing presentation formats for multimedia in order to abstract from these formats to obtain an abstract multimedia content representation model. Once determined such an abstract multimedia content representation model, it allows to be transformed into the syntax and features of the different concrete multimedia presentation formats.

Thus, in contrast to the research activities so far, where usually the ideas and requirements in regard of the desired multimedia composition functionality result in a proper document model, we have performed a backwards analysis. We start at the state-of-the-art of today's presentation formats to obtain the required char-

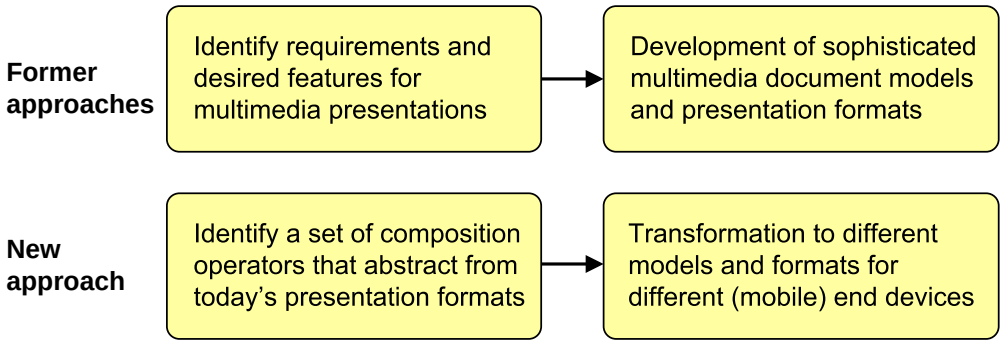


Figure 6.8: Approach for determining the internal multimedia content representation model

acteristics and multimedia composition features of an abstract multimedia content representation model that is independent of the syntax and features of the concrete presentation formats [SB05d].

Research approaches for the sophisticated modeling of multimedia presentations such as Madeus and AHM (see Section 3.2.1) have been contributing to the different facets of composing complex multimedia content but are less significant for this work, since there is no or only limited multimedia player support for the different end user devices today. Research projects that work on abstract internal models are, e. g., the Cuypers engine and WAM project (see Sections 3.2.7 and 3.2.8). The Cuypers engine focuses on high level modeling and sophisticated creation of personalized multimedia content. However, it remains with the delivery of SMIL content for Desktop PCs for the final presentation. The WAM project works on transforming XML-based multimedia content into SMIL with the specific focus of negotiating and meeting the technical constraints of the client device. Supporting a variety of different platforms and formats is not in their focus. The work in [KW06] proposes the Extensible MPEG-4 Textual (XMT) format [ISO01a] for cross-standard interoperability and remains in the field of SMIL-near multimedia document models. Cross-standard transformation among different standards cannot be done lossless and favors often one of the standards, here, SMIL.

6.3.1 Analysis of Existing Presentation Formats for Central Multimedia Features

As introduced above, we pursue in this work an analysis of existing multimedia presentation formats in order to find a greatest common denominator that joins the format-spanning central aspects of the different multimedia presentation formats. As multimedia applications today can use many different presentation formats to display their multimedia content on the user's end device, we selected a wide range of state-of-the-art presentation formats for our analysis. The analyzed presentation formats include SMIL 2.0 and SMIL 2.0 in the Basic Language Profile (BLP) for mobile devices [ABC+01b]. We also considered SVG 1.2 [AAA+04b] and Mobile SVG 1.2 [AAA+04a] comprising SVG Tiny for multimedia-ready cell phones and

SVG Basic for pocket computers like PDAs. The further, we analyzed the formats Flash [Mac04], MPEG-4's XMT-Ω [KWC00, KW06, ISO01a], which is an extension of SMIL aimed as high-level description format for MPEG-4 [KWC00], HTML+TIME [SYS98], and the 3GPP SMIL Language Profile [3rd03a], which is a subset of SMIL targeted for the Multimedia Messaging Service (MMS). The 3GPP SMIL Language Profile is a superset of the SMIL 2.0 Basic Language Profile and is used for scene description within the MMS interchange format [3rd03b].

The earlier comparison and analysis of multimedia document models by Boll et al. [BKW99b] had its focus on finding specific compositional features for multimedia modeling rather than finding an abstract content model. However, in this former analysis and comparison of multimedia presentation formats and document models, the central aspects of multimedia modeling were identified (see Section 3.1.2) that are important in regard of an analysis that aims at finding an abstract multimedia content representation model. These central aspects are among others the definition of the temporal course and visual layout of the multimedia presentation as well as the interaction possibilities provided by it. Within the following analysis, we consider in detail how the syntactic elements of the presentation formats support each of these aspects. Embedding the central characteristics of multimedia modeling, the abstract multimedia content model allows for the transformation into all the different final presentation formats. In the following Sections 6.3.1.1 to 6.3.1.3, the results of the analysis in regard of the temporal model, the spatial model, and the interaction possibilities of the formats are presented. The final section summarizes these results and gives some further conclusions.

6.3.1.1 Analysis of Temporal Models in Multimedia Presentation Formats

The temporal model allows to define the presentation's temporal course, i. e., temporal dependencies between the media elements within a multimedia document (see Section 3.1.2). For this, the media elements need to be arranged along the presentation's timeline [GT95]. The arrangement of the media elements into the presentation's timeline can be conducted by, e. g., determining the single media elements' presentation intervals, employing the specification of timing events, and other approaches. Although, the methods applied for determining the playback time of the media elements can be different, the result is always a playback of the media elements for a specific period of time. The presentation of a medium element begins at a specific point in time within the presentation's playback and lasts for a specific duration. Abstracting from the different methods for determining the playback time of media elements as well as abstracting from the syntax of the different presentation formats, one can see that there are in principal two concepts followed how to model the temporal course of a multimedia presentation.

As depicted in Figure 6.9 by a small slideshow example, the temporal model can be realized by defining absolute positioning in time on a global timeline or by using local timelines (cf. [GT95]). In the first case, one single, global timeline is used as basis to define the temporal synchronization of the media elements (cf. [BKCD98]). This means that all media elements used for the slideshow are placed on the global timeline by distinct points in time, which always refer to the zero point of the global time axis. This concept is realized, e. g., by the SVG family.

The other possibility for temporal positioning of media elements is by using multiple local timelines (cf. [BKCD98]). Here, the media elements refer no longer to the zero point of the global timeline but to the beginning of a local timeline. Local timelines can be nested to build arbitrary relative temporal dependencies between the presentation's media elements. In our example, each slide has its own local timeline where the corresponding image and text elements are placed. These local timelines are then arranged on the global one. The concept of local timelines is supported, e. g., by SMIL and XMT-Ω.

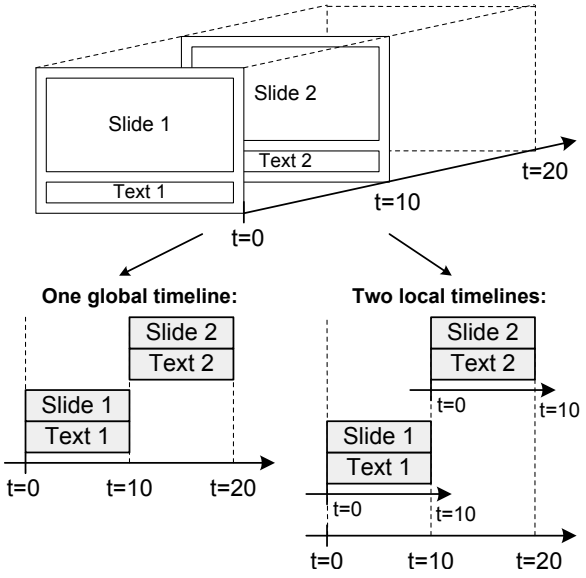


Figure 6.9: Example of a slideshow to illustrate global and local points in time

6.3.1.2 Analysis of Spatial Models in Multimedia Presentation Formats

The spatial model expresses the arrangement and style of the visual media elements in the multimedia presentation, i. e., it determines the spatial layout of the presentation (see Section 3.1.2). Analyzing the syntax of the different presentation formats, we can see that they allow both absolute positioning of visual media elements with respect to the origin of the coordinate system as well as relative positioning of visual media elements at positions relative to other multimedia composition elements. The difference between the two is depicted in Figure 6.10. The top of the figure shows a schematic diagram of a sightseeing application and the visual media elements that are used for it. In the case of absolute positioning the visual media elements always refer to the origin of the coordinate system of the entire multimedia presentation. With relative positioning visual media elements are placed within local areas. In the example given, the medium "Sight's name" and "Image of sight" belong to the coordinate system of the "Info panel". These logical areas can be nested to build arbitrary spatial dependencies between the visual media elements.

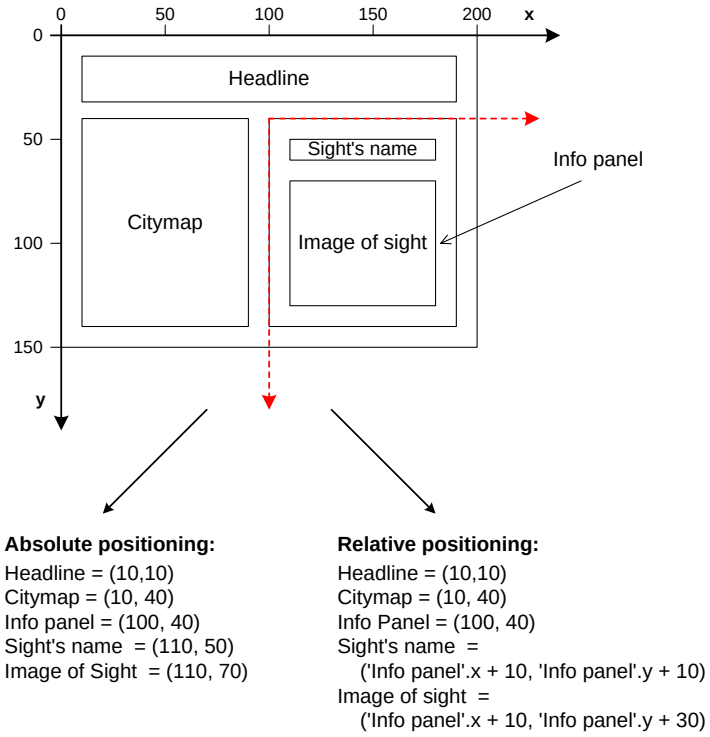


Figure 6.10: Example of a sightsseing application to illustrate absolute and relative positioning in space

6.3.1.3 Analysis of Interaction Possibilities in Multimedia Presentation Formats

The third central aspect in multimedia presentations is interaction. Here, navigational interaction provides for control of the flow of a presentation. It allows to select one out of many presentation paths that can be followed (see Section 3.1.2). This interaction type is realized by, e. g., interactive hyperlinks, menus, and hotspots. Common to these different elements is that they represent one or more interactive external or internal links. External links refer to external multimedia presentations or to a particular medium element within. Internal links, however, refer to a distinct element within the same presentation. Following an internal link means to jump to a specific point in time on the multimedia presentation's timeline. Therefore, they are a proper means to create interactive multimedia presentations in a single document.

All considered presentation formats support unidirectional internal links and unidirectional external links. This means that they provide an element to define a link to a distinct target, within the same document or to other presentations. A link back from the target to the source is thereby not provided (that would be bidirectional linking).

In the following, the definition of navigational links is also called the interaction model of the multimedia content. Such an interaction model allows for determining the possible user interaction with the multimedia presentation in order to let the user choose different playback paths of a presentation.

Although all presentation formats define the considered internal and external navigational interaction, the reality shows something different: For example, not all multimedia players support the internal navigational interaction defined by the standards, i. e., the player does not support jumping to the presentation time of a particular element within the presentation. This is the case, e. g., with Adobe's SVG plug-in [Ado06a] and the SVG 1.2 presentation format. In addition, besides the support for calling external presentations by using external navigational interaction, the current multimedia players do not support for jumping to a particular element within the external presentation, e. g., the RealPlayer [Rea06] displaying a SMIL 2.0 presentation. Finally, not all of the considered multimedia presentation formats define a mechanism for accessing a specific part within a medium element, e. g., the SVG 1.2 format. Here, it would be interesting to provide support to jump at a specific point in time within the playback of a medium element, e. g., a video element, which is used for the presentation. Work in the direction of accessing specific parts of a single medium element is conducted by the Continuous Media Web (CMWeb) project and the Annodex technology [CSI06].

6.3.1.4 Summary of Presentation Format Characteristics

The results of our analysis of the considered presentation formats and their individual characteristics in regard of the discussed temporal, spatial, and interaction models are shown in Table 6.1. One can see that most presentation formats support local timelines to determine the temporal model. Only SVG and its profiles provide a temporal model based on a global timeline. Also most of the presentation formats support relative positioning of media elements in space. Only the profiles of SMIL aimed for mobile devices and Adobe's Flash format use absolute positioning of media elements. In regard of the interaction model all formats support external and internal links. This distinction becomes important when actually generating the multimedia content in final presentation format.

This analysis of the existing presentation formats is targeted to develop a single, presentation-format independent multimedia content representation model. This model is optimally suited for transformation to all the different presentation formats. The design of the internal multimedia content representation model is introduced in the next section.

6.3.2 Definition of the Modeling Characteristics of the Internal Model

For the representation of multimedia content one can find the most different notations, e. g., petri nets in [WSDSS96, BM95, LG90b], flow chart like diagrams with decision operators in [Ado05b, BKCD98], scenario views following the temporal axis paradigm in [JLR+98], activity diagrams with an own extension for the visual layout in [HBR94], hyperchart diagrams which are an extension of statechart diagrams

Presentation format	Temporal model	Spatial model	Navigational interaction
SMIL 2.0	local	relative	internal and external
SMIL 2.0 BLP	local	absolute	internal and external
SGPP SMIL 1.0.0	local	absolute	internal and external
SVG 1.2	global	relative	internal and external
SVG Basic	global	relative	internal and external
SVG Tiny	global	relative	internal and external
XMT-Ω	local	relative	internal and external
Flash	local	absolute	internal and external
HTML+TIME	local	relative	internal and external

Table 6.1: Characteristics of different multimedia presentation formats

in regard of describing time sequencing and synchronization requirements for multimedia in [PTOM98], or tree-like notations in [Bul95]. Since many of today's presentation formats like SVG, SMIL, and XMT-Ω apply XML for their specification and documents based on XML can be represented in a tree structure, a document tree model seems to be best suitable and most intuitive to represent multimedia content. For the development of our abstract multimedia model first the concrete characteristics of the model must be determined to meet the characteristics of the different presentation formats. These are the definition of the temporal, spatial, and interaction model. From the analysis of the presentation formats in Section 6.3.1, we conclude that our abstract document model must provide a temporal model that allows to define local timelines and a spatial model that supports relative positioning. These are requirements that in the end call for a tree structure for our multimedia content representation model. In addition, a tree structure forms a good basis for transforming into the syntax of the often XML-based presentation formats. Our abstract multimedia model is introduced in more detail in the following sections along the central aspects of multimedia document models. We motivate our design decisions by concrete examples.

6.3.2.1 Definition of the Temporal Model

The temporal model of our abstract multimedia content representation model has to be chosen such that we can best generate local and global timelines to support the temporal model of the different presentation formats. A temporal model that bases on local timelines can be directly mapped to the temporal model of a presentation format that also supports local timelines. If the presentation format supports only a global timeline, these local timelines can be automatically transformed into global

positions in time. If we used a global timeline for our abstract model, we could not sufficiently support the creation of semantically reasonable local timelines. Consequently, we decided for our abstract document model in favor of a temporal model with local timelines. These local timelines can be created by defining nested time-intervals as it can be found in, e. g., [DK95]. For the representation of local timelines a tree structure as can be found with SMIL and XMT-Ω forms a reasonable representation. The multimedia document tree represents the time from its left to its right nodes which can be converted to a global timeline.

6.3.2.2 Definition of the Spatial Model

For our abstract multimedia model a spatial model has to be chosen such that it can arrange visual media elements by relative and absolute positioning to provide best support for the different presentation formats' spatial models. A spatial model that supports relative positioning can be directly mapped to a presentation format whose spatial model also supports relative positioning. Such a relative positioning model can also be transformed into absolute positioning of a global coordinate system, as supported by SVG for example. If we used a single global coordinate system for our abstract model, we could not sufficiently support the semantically reasonable creation of local coordinate systems with relative positioning of the media elements. As a consequence, we decided for our abstract spatial model in favor of relative positioning that uses a hierarchical model for specifying the spatial layout and positioning of the media elements.

6.3.2.3 Definition of the Interaction Model

In regard of navigational interaction, the abstract document model supports the definition of both internal links and external links: With internal links navigational interaction within a single multimedia presentation can be created like it is supported by SMIL for example. With external links navigational interaction between different multimedia presentations is modeled as it is supported by the considered formats. The distinction between internal links and external links is important for the later transformation of the multimedia content in the concrete presentation format. Internal links might be considered elegant and desirable in some cases, because the entire multimedia presentation is created within a single document. However, the target of internal links is embedded in the document at transformation time and cannot be changed subsequently. With external links one can refer to external presentations and distinct elements within them. Presentations referred by external links could also be generated on-the-fly with reflecting the latest user profile information just when the user clicks on the respective link. Providing both internal and external links gives the abstract document model the flexibility which is needed to satisfy the different applications' requirements.

6.3.2.4 Summary

In this section, we have defined the characteristics of the models for determining the temporal course, spatial layout, and interaction possibilities of the multimedia presentation with the user in our internal multimedia content representation model.

In the following Sections 6.3.3 to 6.3.5, the basic, complex, and sophisticated multimedia composition support of this internal representation model are presented. This multimedia composition support serves the requirement for providing a set of basic multimedia composition functionality as well as being extensible in regard of more complex and application-specific multimedia composition and multimedia personalization functionality (see Section 5.2.1).

6.3.3 Basic Multimedia Composition Functionality

Since our multimedia representation model has been designed to support local timelines, relative positioning, and interactive linking, it serves the different central modeling characteristics of today's presentation formats. It provides support for creating arbitrary personalized multimedia content in a tree-like fashion abstracting from the features and syntax of the concrete multimedia presentation formats. For it, the abstract representation model provides a set of application-independent basic composition elements. The basic composition elements provide support for the basic functionality for multimedia composition (see requirements to the multimedia composition in Section 5.2.1) and form the primitive building blocks for assembling and composing the selected media elements into personalized multimedia documents (cf. [GT95]).

The set of basic composition elements can be divided into the following groups: *media elements*, *basic composition operators*, and *projectors*. These groups of basic composition elements are introduced in the following.

6.3.3.1 Media Elements

The MM4U framework provides support for discrete as well as continuous media elements. As introduced and defined in Section 3.1.1, discrete media elements are images and texts and continuous media elements are video clips and audio clips. These are represented in the internal multimedia content model by the media elements Image, Text, Video, and Audio. The duration of discrete media elements is defined per default to *infinite*. Continuous media elements have an intrinsic playback duration, which is given implicitly by the length of their raw media data. In addition, visual media elements such as Image, Text, and Video have a spatial position as well as spatial extension.

6.3.3.2 Basic Composition Operators

A basic composition operator or short a basic operator can be regarded as an atomic unit for multimedia composition, which cannot be further broken down. Basic operators are quite simple. However, they are applicable for arbitrary application areas and therefore most flexible. As depicted in Figure 6.11, the basic composition operators are specialized into three groups: *basic temporal operators*, *basic selector operators*, and *basic interaction operators*. These different types of basic composition operators are introduced in the following.

Basic Temporal Operators The temporal course of a multimedia presentation is determined by basic temporal operators or short temporal operators. The Parallel

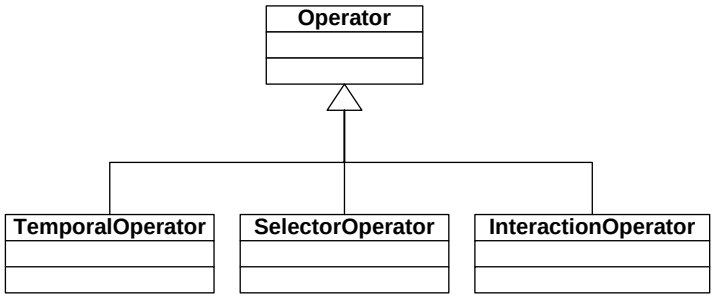


Figure 6.11: The three types of basic composition operators

operator is one specialization of the temporal operator. It is used to present its children, which are either media elements or other multimedia document trees, at the same time. It represents the functionality of the <par> element in presentation formats like SMIL. The Parallel element is not contradictory to presentation formats that only support a global timeline, because in that case the children of the Parallel element would just be placed at the same point in time on the presentation format's global timeline. The corresponding UML class diagram defining the Parallel operator is depicted in Figure 6.12. It has one or more multimedia composition operators or media elements associated. Within the representation model, composition operators and media elements are subsumed under the notion of multimedia composition variables or short composition variables. Thus, as depicted by the UML diagram in Figure 6.12, the media elements and composition operators are specific subtypes of the composition variable. The projectors depicted at the right hand side of Figure 6.12 are used to determine the spatial, typographic, temporal, and acoustic layout of the multimedia presentations. They are introducted later in Section 6.3.3.3.

The basic temporal composition operator Sequential represents the functionality as can be found with the <seq> element of SMIL. The children of the Sequential operator are the same as for the Parallel operator, but the Sequential operator's semantics is to show them in a sequence, one after the other. The corresponding UML class diagram defining the Sequential operator is shown in Figure 6.13.

With the composition operator Delay, a distinct gap within a presentation can be determined, i. e., a period of time where the respective presentation sub-tree pauses (*delays*). As defined in Section 6.3.2.1, the internal multimedia content representation model represent time from its left to its right nodes. Consequently, the Delay operator shifts the subsequent nodes on the right side in the document tree by the determined duration on the timeline. The Delay operator can be placed between any media elements and composition operators and has no child node. An example of the Delay operator in the notation of Z$_Y$X (see Section 3.2.1) is shown in Figure 6.16. The corresponding UML class diagram defining the Delay operator as well as the class diagrams defining the following temporal composition operators are listed in Appendix B.

The Loop operator has exactly one child node, representing an arbitrary presentation subtree. Its semantics is to repeat the associated presentation subtree for a

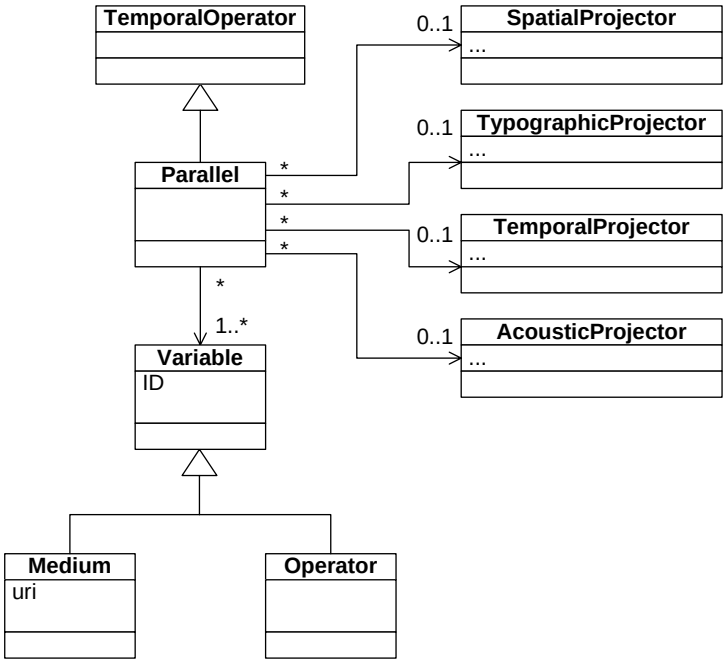


Figure 6.12: The basic temporal operator Parallel

specified times. Consequently, it can be considered as a Sequential operator where the single child nodes are a n-times duplicated presentation subtree and therefore the presentation subtree is presented n-times.

With the Jump operator, a non-interactive link to a specified target can be defined. The Jump operator is automatically activated when the playback of the multimedia presentation reaches the operator. The target of a Jump operator can be a specific presentation element within the current presentation (internal jump) but it can also be an external multimedia presentation or a specific medium element within this external presentation (external jump).

Basic Selector Operators The basic selector operators or short selector operators are used to cut parts of a single medium element or a presentation tree in regard of its temporal or spatial dimension. There are three selector operators defined in our internal multimedia content representation model. These are described in the following.

The TemporalSelector operator is applied to shorten the playback duration of a medium element or multimedia document tree. It realizes the specific temporal multimedia composition functionality that is introduced, e. g., in SMIL with the clipBegin and clipEnd attributes. A TemporalSelector operator cuts a distinct time-interval of the intrinsic presentation duration of a medium element or multimedia document tree. As the presentation duration of the discrete media elements Text and Image is infinite per default, the TemporalSelector is used to shorten it to a finite duration.

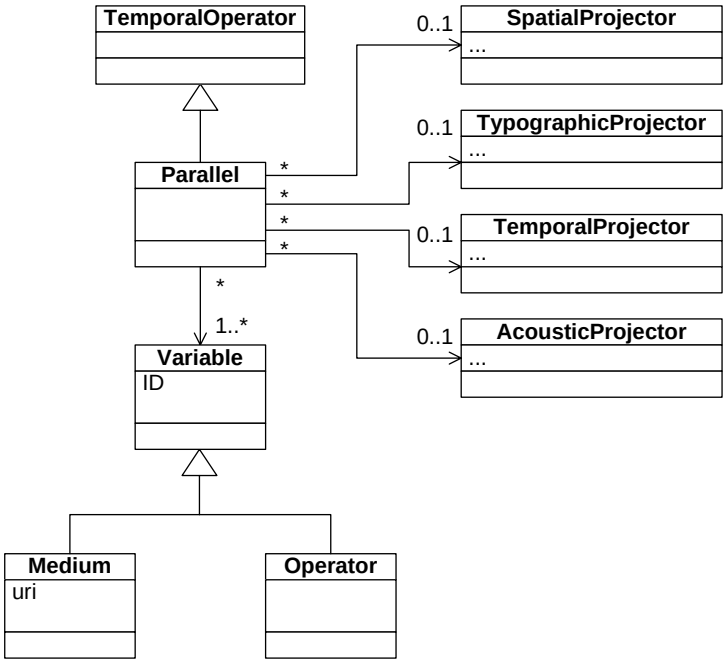


Figure 6.13: The basic temporal operator Sequential

For reasons of clarity the TemporalSelector and the Delay operator abandon the intermix of media element definitions and temporal synchronization as it has been introduced with attributes such as begin into models like SMIL and SVG for reasons of convenience. The UML class diagram defining the TemporalSelector operator is depicted in Figure 6.14.

Like the TemporalSelector operator is used to select a distinct time interval, the SpatialSelector is used to cut a specific spatial region of a single medium element or multimedia document tree. Both, the TemporalSelector and SpatialSelector can be applied recursively. With the TextualSelector a distinct part of a text medium can be cut out and used for composition. However, in contrast to the TemporalSelector and SpatialSelector, a TextualSelector can only be applied to a text element and not be applied recursively. The UML class diagrams of the TemporalSelector and SpatialSelector are shown in Appendix B.

Basic Interaction Operators With basic interaction operators or short interaction operators the navigational interaction possibilities of a multimedia presentation are realized. The internal multimedia representation model defines three interaction operators. These are presented in the following.

A Link operator defines an interactive link—represented by a single medium element or a fragment of a multimedia presentation that is click-able by the user—and a target presentation the user receives if he or she clicks on the link. The corresponding UML class diagram of the Link operator is presented in Figure 6.15. The

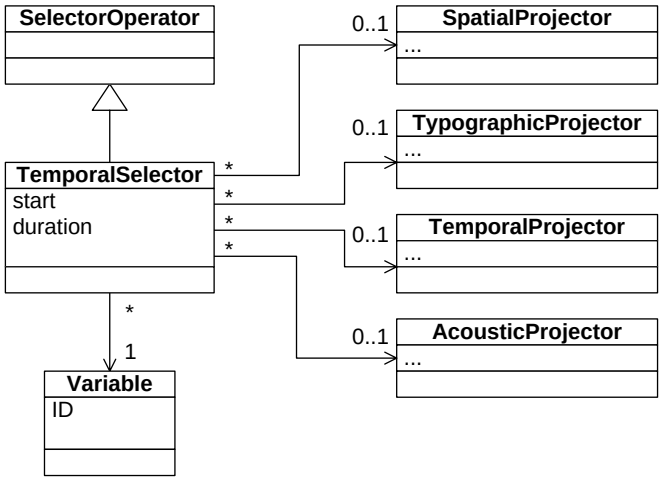


Figure 6.14: The basic selector operator TemporalSelector

target of the Link operator can be an external multimedia presentation represented by a URI. It can also be an internal reference to a specific part of the multimedia presentation identified by a composition variable, i. e., a medium element or composition operator. The Link operator's attribute mode determines what happens to the original presentation when the user clicks on the link to start the target presentation. In the case of mode = vanish, the current presentation is stopped and replaced by the presentation determined by the Link operator's target. If mode = prevail, the playback of the current presentation is continued and simultaneously the presentation defined by the Link operator's target is presented. Here, the current presentation and the target's presentation are "merged" together.

With the Hotspot operator a second type of interaction operator is defined within the internal representation model. Like the Link operator, the Hotspot operator has one child node associated, which is presented when the playback engine reaches the Hotspot operator. However, in contrast to the Link operator, the Hotspot operator has defined multiple targets. Each target is defined by a click-able rectangle area within the Hotspot operator's associated child node, which is like with the Link operator either a medium element or presentation fragment.

With basic composition operators and media elements multimedia presentations can be created such as the slideshow depicted in Figure 6.16. The notation of the tree-like diagram is based on [Bol01, BK01]. The basic composition operators are represented by white rectangles and the media elements by gray ones. The relation between the media elements and the basic operators is shown by the edges beginning with a filled circle at an operator and ending with a filled rhombus respectively a diamond at a media element or another operator. Time is represented within the tree-like diagram from left to right. The diagram at the bottom depicts the local timelines of the slideshow presentation.

The semantics of the slideshow shown in Figure 6.16 is that it starts with the presentation of the root element, which is the Parallel operator. As the semantics of

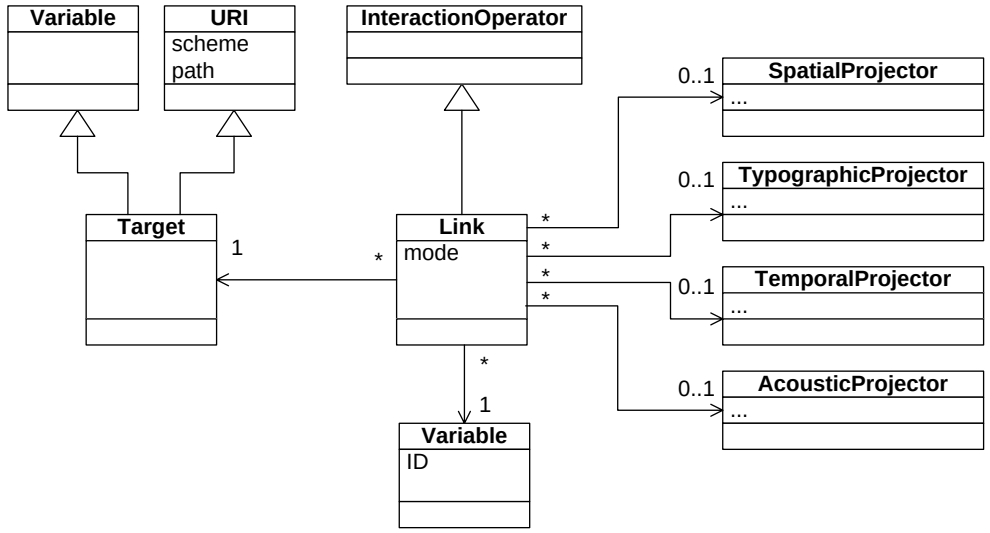


Figure 6.15: The basic interaction operator Link

the Parallel operator is that it shows the operators and media elements at the same time, the audio file starts to play while simultaneously the Sequential operator is presented. The semantics of the Sequential operator is to show the attached media elements one after another, so while the audio file is played in background the three slides are presented in sequence after an initial delay of five seconds due to the Delay operator. If the user clicks on a slide, the presentation immediately jumps to the Link operator's target, i. e., the next slide. When the slideshow presentation is finished, the Jump operator is activated and the presentation continues with the external multimedia presentation.

As the slideshow example shows, each temporal composition operator defines its own local timeline for determining the global temporal synchronization of the presentation. For example, the timeline of the audio medium begins at $0s$ and ends at $35s$, while the local timeline of the second slide image begins at $15s$ and ends with $25s$. Since the temporal composition elements can be nested, multimedia presentations of arbitrary temporal layout can be created with the basic multimedia composition elements.

6.3.3.3 Projectors

Besides the basic composition operators also so-called projectors are part of basic multimedia composition functionality the representation model provides. Projectors can be attached to media elements and composition operators to define, e. g., the visual layout and acoustical layout of the multimedia presentation. As shown in Figure 6.17, there exist four different types of projectors.

The SpatialProjectors are used to determine the relative spatial positioning as well as the width and height of visual media elements and multimedia document

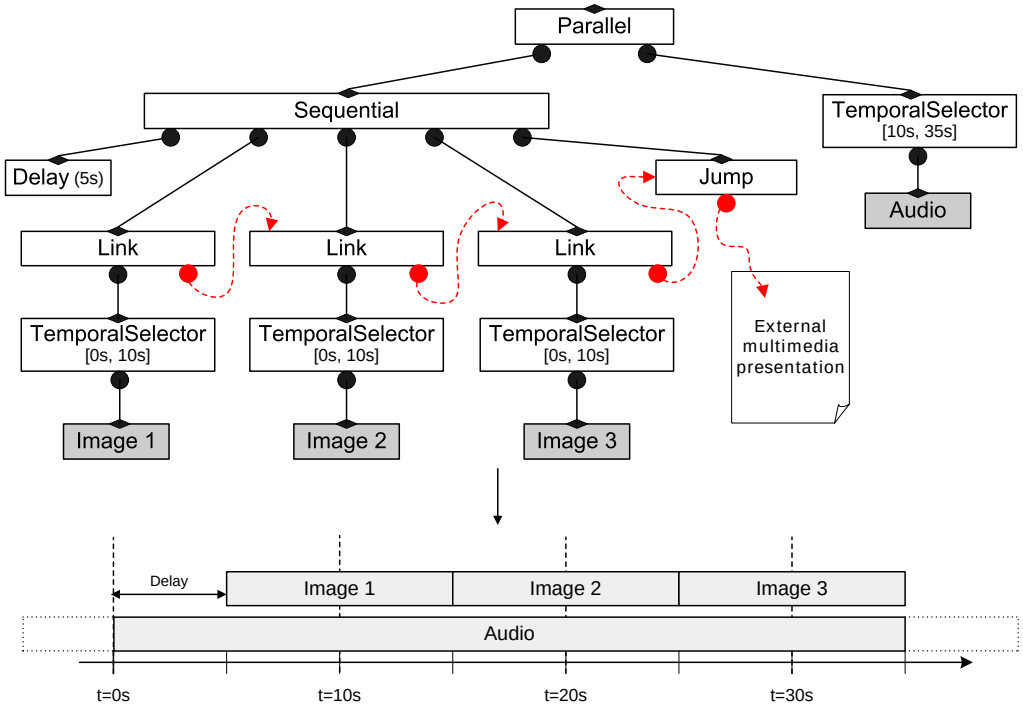


Figure 6.16: Slideshow as an example of assembled multimedia content

sub-trees. They realize the visual layout of the multimedia presentation following the *region*-concept of SMIL and embracing the hierarchically assignment of layout to media elements in SVG, respectively. The positioning as well as width and height of the spatial projection can be provided in different units such as cm, inch, and pixel. Default value for the unit attribute is pixel. In addition, the z-order value can be determined with the priority attribute. It is required, e. g., to determine the presentation order of two partially overlapping image elements.

The acoustic layout of the multimedia presentation is defined by attaching an AcousticProjector to an audio element or video element. With it, the volume and balance of an audio medium or presentation sub-tree can be determined.

Figure 6.18 shows a slightly enhanced version of the slideshow example from above with projectors attached (the Link and Jump operators as well as the image elements' and audio element's temporal selectors are not depicted for reasons of simplicity). As the figure shows, each SpatialProjector defines its own (local) coordinate system within the visual media elements can be placed. For each SpatialProjector the diagram at the bottom of Figure 6.18 shows its spatial extension in the presentation. As composition operators can be nested, also the projectors can and influence each other. For example, the projectors marked with 3 place the slide's image elements into the local coordinate system that is spawn by projector 2. Consequently,

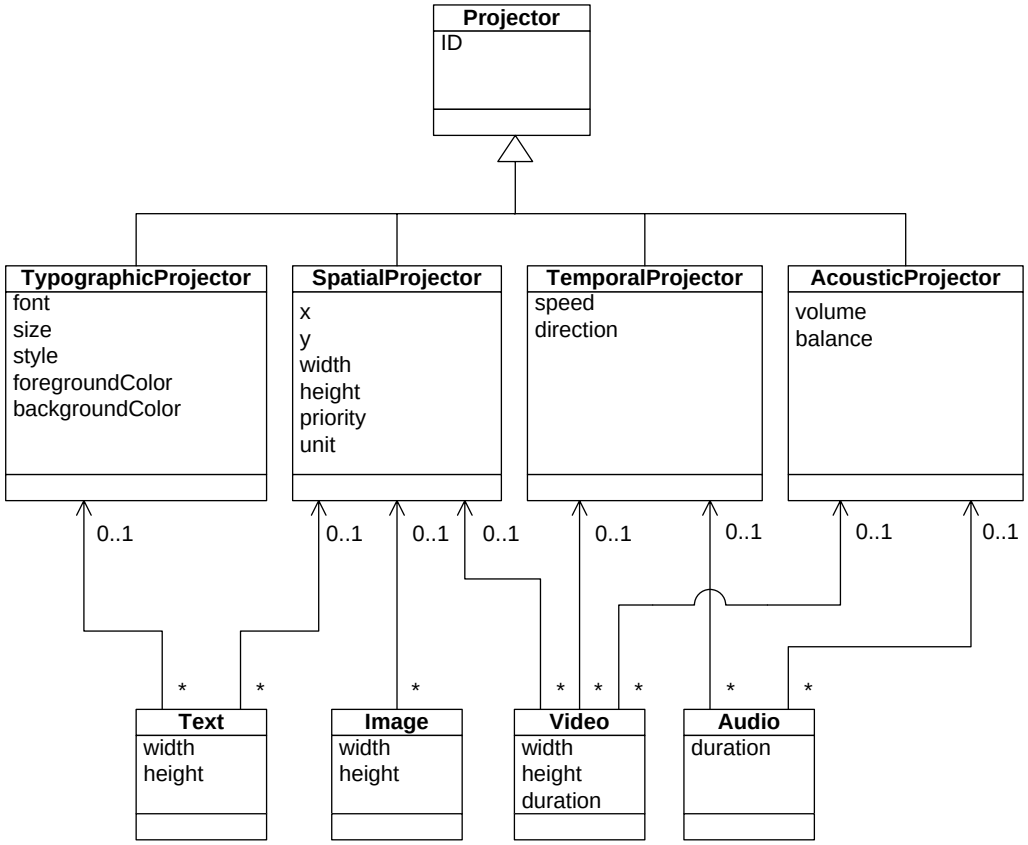


Figure 6.17: The projectors of the internal multimedia content representation model

with nesting SpatialProjectors multimedia presentations of arbitrary hierarchically organized spatial layout can be realized.

As Figure 6.17 shows, the internal representation model defines besides the SpatialProjector and AcousticProjector two other projectors. With the TemporalProjector the temporal course of a presentation can be modified in regard of the playback speed and the playback direction. When two or more TemporalProjector are used within a multimedia presentation tree, the values of the speed attribute are appropriately accumulated. For example, if one TemporalProjector determines the playback speed to be 2 and another TemporalProjector in the presentation sub-tree determines the playback speed to be 3, the total playback speed of the multimedia presentation sub-tree associated under the second TemporalProjector is $2 \times 3 = 6$.

With the TypographicProjector, the typographic layout of text elements can be modified. This regards typographic attributes such as the used font, font size, font style including bold, italics, and the like, as well as text color and background color.

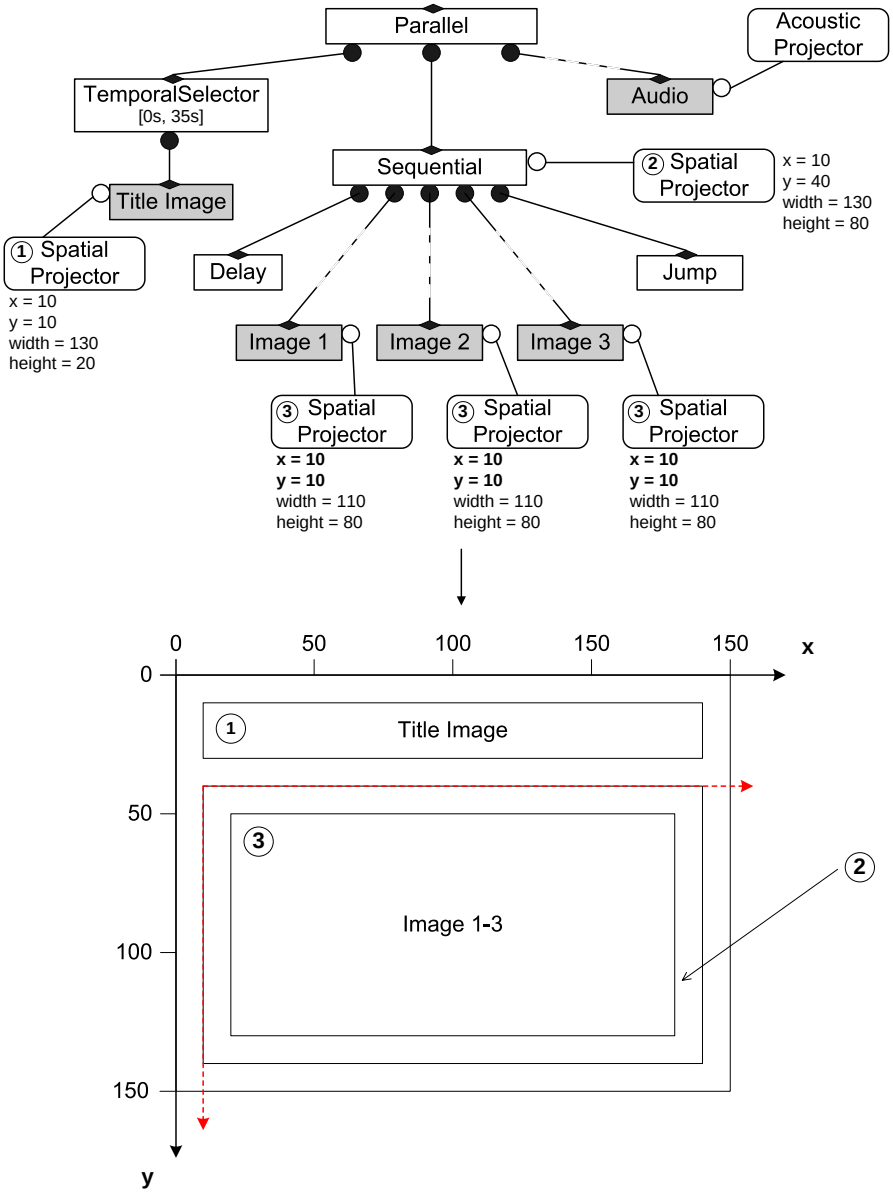


Figure 6.18: Adding layout to the slideshow example

As Figure 6.17 shows, not all projectors are reasonable for all the four media elements defined in our internal multimedia content representation model. The reasonable relations between media elements and projectors are defined by the directed associations from the media elements to the projectors.

6.3.3.4 Summary

The description above presents the basic multimedia composition functionality the internal multimedia content representation model offers. It provides customary composition operators for creating multimedia content as provided by modern multimedia presentation formats like SMIL and SVG. Even though the basic composition functionality does not reflect the fancy features of some of today's multimedia presentation formats, it supports the very central multimedia features of modeling time, space, and interaction. This allows the transformation of the internal multimedia content representation model into many different multimedia presentation formats for different end devices.

With the media elements, the basic composition operators, and the projectors, the multimedia content representation model provides the basic bricks for composing and assembling arbitrary personalized multimedia content. However, so far the MM4U framework provides "just" basic multimedia composition functionality. In the same way as one would use an authoring tool to create SMIL presentations, for example the GR*i*NS editor [Ora06c], one can also use a corresponding authoring tool for the basic composition operators the MM4U framework offers to create multimedia content. Such an authoring tool for the internal multimedia content representation model exists with the context-driven smart authoring environment *x*SMART presented in Section 10.6. For reasons of reusing parts of the created multimedia presentations, e. g., parts of the presentation's layout, and for convenience, there is a need for more complex and application-specific multimedia composition functionality for creating personalized multimedia content. Consequently, we show in the subsequent Section 6.3.4 how this basic composition functionality the internal representation model provides can be extended by more complex and application-specific multimedia personalization support.

6.3.4 Complex Multimedia Composition Functionality

For creating more complex multimedia presentations, the internal multimedia content representation model provides the ability to abstract from the set of basic composition elements to more complex operators. A complex composition operator or complex operator encapsulates the composition functionality of an arbitrary number of media elements, basic operators, and projectors. Employing such complex operators, application developers can create more complex bricks for realizing their multimedia composition functionality. Besides encapsulating media elements, basic operators, and projectors, complex composition operators can also contain other complex operators. Consequently, complex composition operators provide a means for reusing composition operators and parts of multimedia presentations, respectively. Thus, the abstract multimedia content representation model provides with the complex composition operators for the requirement to provide for more complex multimedia composition functionality (see Section 5.2.1). The UML diagram in Figure 6.19 shows this functionality. A complex operator can embed an arbitrary number of media elements, basic operators, and projectors, as well as it can include an arbitrary number of other complex operators.

In contrast to the basic operators, the complex composition operators can be dismantled into their individual parts. Figure 6.20 depicts a complex composition

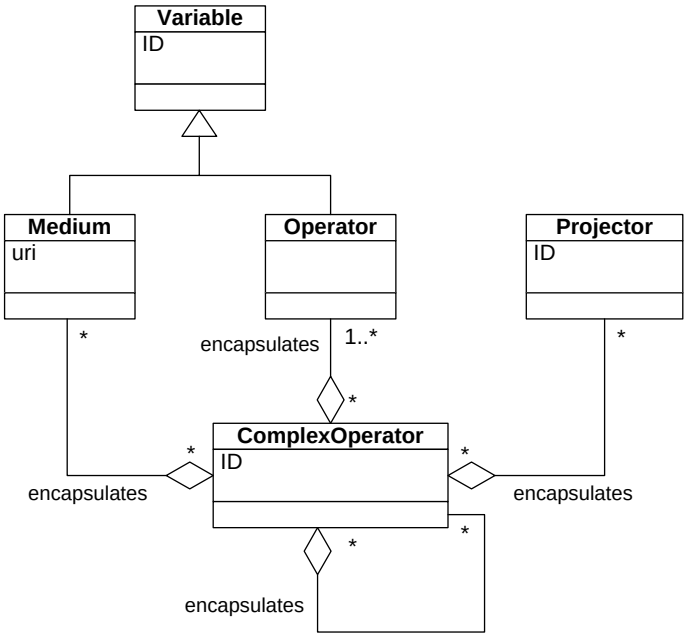


Figure 6.19: The concept of complex composition operators in the internal multimedia content representation model

operator for our slideshow example above (the temporal selectors as well as the projectors are not shown for reasons of simplicity). The complex composition operator encapsulates the media elements, operators, and projectors of the slideshow (the latter are omitted in the diagram to reduce complexity). The complex operator Slideshow, indicated by a small "c"-symbol in the upper right corner, represents an encapsulation of the former slideshow presentation in a complex object and forms itself a building block for more complex multimedia composition.

Complex operators as described above define fixed encapsulated presentations. Their temporal course, spatial layout, and the used media elements cannot be changed subsequently. However, a complex composition operator does not necessarily need to specify all media elements, operators, and projectors of the respective multimedia document tree. Instead, to be more flexible, some parts can be intentionally left open. These parts constitute the parameters of a complex composition operator and have to be filled in for concrete usage of these operators. Such parameterized complex composition operators or short parameterized complex operators are one means to define multimedia composition templates within the MM4U framework. However, only pre-structured multimedia content can be created with these templates, since the complex composition operators can only encapsulate presentations of a fixed structure.

Figure 6.21 shows the slideshow example as a parameterized complex composition operator. In this case, the complex operator Slideshow comprises the two basic operators Parallel and Sequential. The Slideshow's parameters are the place holders

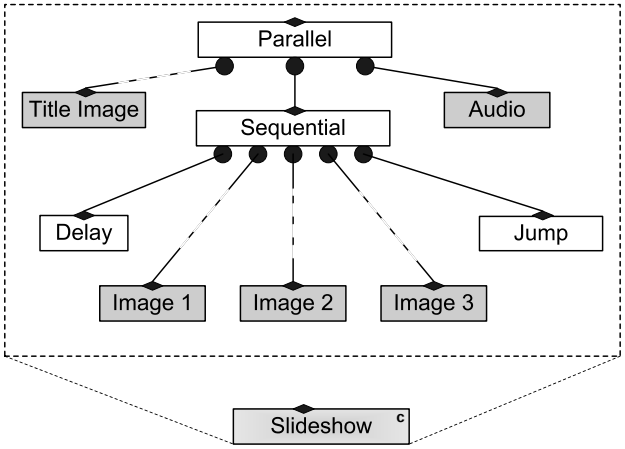


Figure 6.20: The slideshow example as a complex composition operator

for the title screen, Delay operator, three image elements "Image 1" to "Image 3", and the Jump operator. The slideshow's audio file is already preselected and encapsulated by the complex operator.

In addition, the parameters of a complex composition operator can be typed, i. e., they expect a special type of operator or media element. The Slideshow operator would expect an image element for the title screen, followed by a Delay operator, three slide images, and a Jump operator to a subsequent presentation. However, the Slideshow operator could also be configured such that it requires an image element as title screen and arbitrary media elements or multimedia composition operators for the five following parameters, each determining an individual slide. To indicate the complex operator's parameters, they are visualized by rectangles with dotted lines in Figure 6.21. Only by providing concrete media elements for the single slides, the complex operator Slideshow can be used.

In the same way as projectors are attached to basic operators in Section 6.3.3, they can also be attached to complex operators. The SpatialProjector attached to the Slideshow operator shown in Figure 6.21 determines that the slideshow's position within a multimedia presentation is at the position of $x = 100$ pixel and $y = 50$ pixel in relation to the position of its parent node.

With basic composition operators and complex composition operators one can build multimedia composition functionality that is equivalent to the composition functionality of advanced multimedia document models like Madeus [JLR+98] and ZYX [BK01]. Though parameterized complex composition operators can have an arbitrary number of parameters and can be configured individually each time they are used, the internal document structure of complex operators is still *static*, like with the complex composition operators without parameters. This means that the structure of the multimedia presentation is hard-wired within the parameterized complex operator. Once a parameterized complex composition operator is defined, the number of parameters and their type are fixed and cannot be changed. Using a parameterized complex composition operator can be regarded as filling in fixed

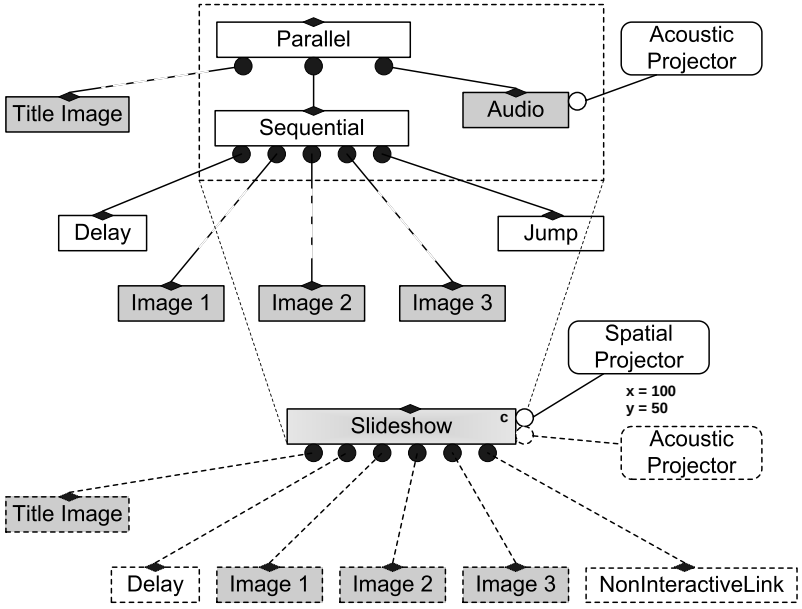


Figure 6.21: The slideshow example as parameterized complex composition operator

composition templates with suitable media elements. Personalization can only take place in selecting those media elements that best fit the user profile information. For the dynamic creation of *personalized* multimedia content, as stated in the requirements in Section 5.2.2, even more sophisticated composition functionality is needed that allows the composition operators to change the structure of the generated multimedia content at runtime. To realize such sophisticated composition functionality additional (application-specific) composition logic needs to be included into the composition operators, which cannot expressed anymore even by the mentioned advanced document models we find in the field. Consequently, the sophisticated operators presented in the following section enhance the concept of complex operators by determining the structure and layout of the multimedia presentation during runtime and the depending on the user profile information.

6.3.5 Sophisticated Multimedia Composition Functionality

With basic and complex composition functionality, we already provide the dynamic composition of pre-structured multimedia content by parameterized multimedia composition templates. However, such templates are only flexible concerning the selection of the concrete composition parameters. To achieve an even more flexible dynamic content composition, the framework provides sophisticated composition operators or short sophisticated operators, which allow determining the document structure and layout during creation time by additional composition logic. Multimedia composition templates defined by using such sophisticated composition

operators are no longer limited to create pre-structured multimedia content only. However, they allow, e. g., to determine the concrete temporal course, the document structure, spatial and acoustic layout, and the number of used media elements for the multimedia presentation on-the-fly and depending on the user profile information and any further additional information such as a database containing sightseeing information.

Such sophisticated composition operators exploit the basic and complex composition functionality. In addition, they allow more flexible, possibly application-specific multimedia composition and personalization functionality with their additional composition logic. Thus, the sophisticated composition operators serve the requirement for providing application-specific multimedia personalization functionality (see Section 5.2.1). The provided application-specific multimedia composition logic can be realized by different approaches for multimedia personalization (see Section 3.3) and using, e. g., generic document structures, templates, style sheets, layout constraints and rules, or plain programming code. Consequently, the sophisticated composition operators also serve the requirement for exploiting different personalization approaches (see Section 5.2.1). In our graphical notation, sophisticated composition operators are represented in the same way as complex operators, but are labeled with a small "s"-symbol in the upper right corner. Besides the complex and basic composition functionality, a sophisticated composition operator can also embed other sophisticated composition operators. Like the complex composition operators, also the sophisticated composition operators can be parameterized. Such a composition operator is called parameterized sophisticated composition operator or short parameterized sophisticated operator.

Figure 6.22 shows an example of a parameterized sophisticated composition operator, the SightPresentation. This operator provides the generation of a multimedia presentation about a sight. The parameters of this sophisticated operator are the sophisticated operator SightContent, representing a sequence of image elements or video elements together with either textual descriptions or auditive information about the sight, as well as an optional parameter for determining a background music for the presentation. Optional parameter means that the parameter is not necessary for applying the sophisticated composition operator. If the parameter is provided, the application-logic of the SightPresentation operator integrates the background music. If it is not provided, the operator changes the appropriate part of the sight presentation. The SightPresentation operator is used within our personalized city guide application presented in Section 10.1 and serves there for Desktop PCs as well as mobile devices.

The multimedia document tree generated by the SightPresentation operator is shown in the bottom part of Figure 6.22. Its root element constitutes the Parallel operator. Attached to it are the title image of the sight presentation and a Link operator for returning to the tourist guide's main page, which is a map of the city with a personalized selection of spots on it. Also attached to the Parallel operator is another Link operator referring to the sight's homepage (if applicable) and the sophisticated operator BackgroundMusic. The sophisticated operator BackgroundMusic is carrying the actual sight presentation content in form of the sophisticated operator SightContent as well as a piece of music. Thus, the optional parameter determining a background music is provided in this case.

At this point it is important to note, that independent of how the sophisticated multimedia content composition functionality is actually realized and independent of the input parameters given to such a sophisticated operator, the result of this composition process is always a multimedia content representation tree that only consists of basic multimedia composition elements, i. e., media elements, basic operators, as well as projectors as introduced before. This holds also if complex operators are used within a sophisticated operator, since these can be dismantled into their basic composition bricks. Consequently, in regard of specifying the sophisticated operators' functionality, the UML class diagram depicted in Figure 6.19 also holds for this kind of multimedia composition support. No additional specification is required.

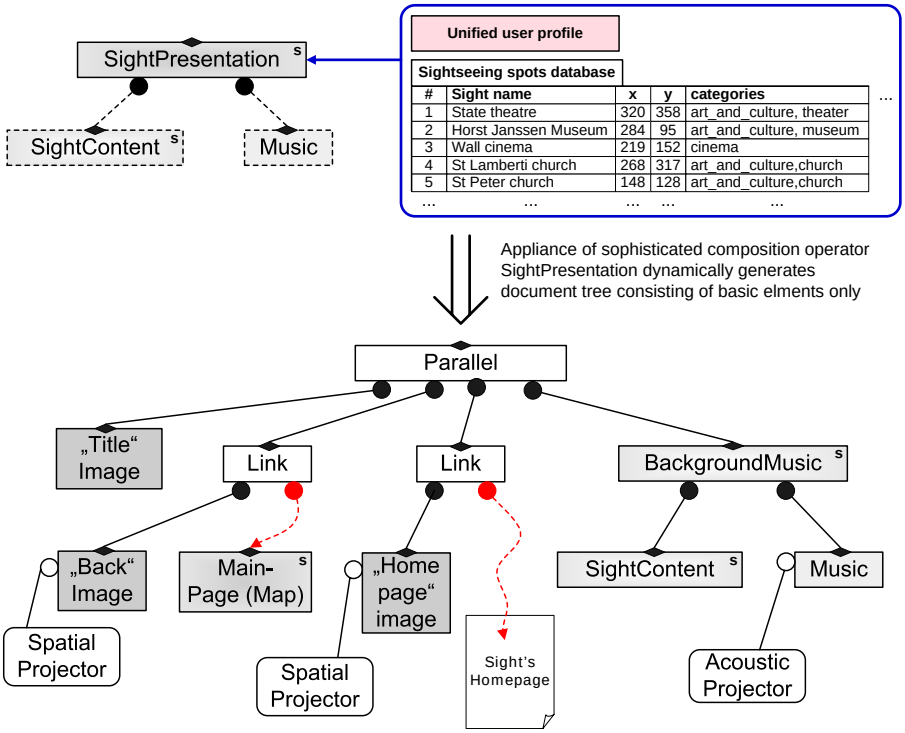


Figure 6.22: Insight into the sophisticated composition operator SightPresentation

The SightPresentation operator is one example of extending the multimedia composition and personalization functionality of the Multimedia Composition layer by a sophisticated application-specific multimedia composition operator, here in the area of (mobile) tourism applications. This operator, e. g., is developed by programming the required dynamic multimedia composition functionality. However, the realization of the internal composition logic of sophisticated operators is independent of the used technology and programming language. The same composition logic could also be realized by using a different technology, as for example a template or constraint-based approach. Though the actual realization of the SightPresenta-

tion operator would be different, the multimedia document tree generated by this personalization approach would be the same as depicted in Figure 6.22. Two example result of applying the sophisticated operator SightPresentation are depicted in Figure 6.23. The left hand side of the figure shows a sight presentation about a museum targeted at a Desktop PC, comprising video elements and text elements, however, no auditive information. The right hand side of the figure shows the application of the same sophisticated operator SightPresentation, however, with different parameters provided. The result is a sight presentation targeted at a mobile device and carrying a slideshow of images and some auditive information about a restaurant.

Figure 6.23: Results of applying the sophisticated composition operator SightPresentation

Sophisticated composition operators allow embracing the most different solutions for realizing personalized multimedia composition functionality. This can be plain programming as in the example of the SightPresentation operator, document structures and templates that are dynamically selected according to the user profile information and filled in with media elements relevant to the user, or systems describing the multimedia content generation by constraints and rules. As stated above, it is important that the result of applying such a sophisticated composition operator is always a multimedia document tree that consists (dismantled to its lowest units) only of a basic multimedia composition elements, i. e., media elements, basic operators, and projectors. Thus, the personalization of the multimedia content takes place *within* the sophisticated composition operator and not with the resulting document tree.

Our multimedia content representation model with its basic composition elements provides the basis needed to develop arbitrary sophisticated multimedia composition and personalization functionality. Thus, sophisticated composition operators can be seen as a "small personalized multimedia application" itself that can conduct a particular multimedia personalization functionality. This small application can be re-used within others and thus extends the functionality of the framework. With the sophisticated composition operators the Multimedia Composition layer provides its most powerful and flexible functionality to generate arbitrary personalized multimedia content.

6.3.6 Alternative Representations of the Internal Model

Besides the presented semi-formal specification of the internal multimedia content representation model by means of UML diagrams, the internal representation model is also fully specified in the enhanced entity-relationship model (EER model) [EN03]. The EER model also allows for defining the presented basic, complex, and sophisticated multimedia composition functionality. However, for reasons of uniformity with the presentation of the unified user model and the software engineering figures basing on UML, we decided to favor the specification in UML rather using the EER model.

In addition, also steps have been undertaken to provide a formal specification of the internal representation model. Here, the textual Object Constraint Language (OCL) of the UML is applied. The OCL allows for specifying additional constraints on UML models in a more precise and concise way than it is possible with the graphical means provided by UML [Ric02]. Richters [Ric02] has specified a formal syntax and semantics of types, operations, expressions, invariants, as well as pre-/postconditions of OCL. Together with an appropriate tool support called USE [Uni06], it is possible to analyze and validate UML models and OCL constraints. The benefit that can be achieved with such a formal specification of the internal multimedia content representation model compared to UML and EER is that the recursive structures and dependencies of the multimedia content tree can be fully captured. With adding OCL constraints to the UML model one can check with the USE tool constraints on the recursive structure of the multimedia content tree. For example, constraints can be specified that allow to determine whether all media elements have an appropriate projector in the composition tree. For example, an image element needs at least one SpatialProjector along the path from the medium element to the root node of the composition tree. In addition, a vice versa constraint can be specified to guarantee that a projector in the multimedia document tree has effect on at least one medium element in its subtree. For the internal multimedia content representation model, we specified among others the constraints above and evaluated them with the USE tool. What we observed is that not surprisingly the efforts are very high for providing a formal specification of the representation model. Thus, we decided in favor of the semi-formal specification of the representation model as presented above. In cases where, e. g., the constraint for having at least one appropriate projector per medium is violated, we assume that default values are applied instead. For example, if a spatial projector is missing for an image element, the default position of the image element is at $x = 0$ pixel and $y = 0$ pixel. The de-

fault width and height of the image element is the width and height provided by the image element's media data.

6.3.7 Summary

With the internal multimedia content representation model, we introduced different level of multimedia composition support: the basic, complex, and sophisticated multimedia composition functionality. This multimedia composition functionality is reflected by the UML diagram in Figure 6.19. Besides the relation of the basic composition elements such as media elements, basic composition operators, and projectors to the complex operators including complex and sophisticated multimedia composition functionality, the diagram also shows that the composition elements in the internal representation model each have a unique ID. This allows for identifying the composition elements within the multimedia content representation tree. For modeling these IDs, e. g., the concept of DeweyIDs [HHMW06] can be applied allowing for uniquely referring nodes in XML documents.

Applying the basic, complex, and sophisticated multimedia composition functionality, the multimedia content is represented in an abstract representation model and has to be transformed into a presentation format that can be rendered and displayed by multimedia players on the end devices. This transformation to the final multimedia presentation formats is described in the following section.

6.4 Multi-Channel Multimedia Content Transformation

The personalized multimedia content authored in the internal multimedia content representation model is transformed by applying a multi-channel transformation approach into standardized multimedia presentation formats targeted at different (mobile) end devices (cf. [BH05, Top02]). Thus, we serve the requirement to use and rely on existing multimedia presentation software (see Section 5.2.1) to render and display the generated presentation on the (mobile) client device.

The multi-channel transformation of the internal multimedia content representation model consists of different tasks resulting from the different characteristics of the target formats that have to be considered: how is the temporal and spatial model defined, what interaction possibilities are provided, and which syntactic elements does the targeted format support to actually realize the multimedia composition functionality? For each of the target options, we discuss in the following how they are realized in our MM4U framework.

If the targeted presentation format's temporal model supports local timelines, then the abstract content tree already holds the time model of the targeted format. Analogously, if the target format supports relative spatial positioning, then the abstract content model already reflects the spatial model of the presentation format. If the presentation format supports both internal and external linking, then the abstract content model already reflects its interaction model. In any other case the temporal, spatial, or interaction model have to be adapted to the features of the respective model(s) of the concrete presentation format. For example, the transformation from local timelines to a global timeline is needed for target formats such as

the SVG family and is presented in Section 6.4.1. The transformation from relative positioning to absolute positioning as needed, e. g., for SMIL 2.0 Basic Language Profile (BLP), is presented in Section 6.4.2. The transformation of the interaction model to the features of a concrete presentation format is described in Section 6.4.3. Finally, the actual mapping of the abstract content representation model to the concrete syntax of the target formats by means of transformation rules is described in Section 6.4.4. The transformation rules applied for mapping the basic multimedia composition elements to the syntax of the target formats SMIL, SVG, and Flash are presented in Sections 6.4.5 to 6.4.7.

At this point, it is important to note that from one abstract multimedia document (in the internal multimedia content representation model) not necessarily all of the different presentation formats are reasonable for transformation. For example, if the multimedia document is designed for a Desktop PC with a minimum screen resolution of 1024×768, reasonable target formats are SMIL 2.0, SVG 1.2, and Flash. To present the same multimedia document on a mobile device by using Mobile SVG or SMIL 2.0 BLP, typically media assets of smaller resolution have to be selected and the spatial layout of the presentation has to be changed. This is supported by our multimedia personalization framework with its ability for a personalized and with it end device dependent selection of media elements (via the framework's Media Pool Accessor and Media Data Connectors layers) and its extensibility in regard of integrating arbitrary (possibly application-specific) multimedia composition and multimedia layout functionality via the sophisticated composition operators.

6.4.1 From Local Timelines to a Global Timeline

If the presentation format's temporal model supports local timelines, then the abstract document tree already holds the time model of the targeted format. If the presentation format supports a global timeline, the local timelines are transformed by the algorithm presented in Listing 6.1. The syntax of this algorithm bases on the pseudo-code notation used in [RN03].

The algorithm for transforming the local timelines to a global timeline starts at the root element of the document tree and the zero point in time with the initial call totalDuration <- CalculateGlobalTimeline(rootnode, 0). It traverses the nodes of the document tree from left to right in depth first order. On traversing, the algorithm accumulates the global time of the presentation from the local timelines. The duration value added to the global presentation time depends on the node type. For the node types Medium (which includes Image, Text, Audio, and Video), TemporalSelector, and Delay the type's intrinsic duration is added to the global time (lines 8 to 10). The composition element Sequential itself does not add to the global time. The time progresses by the traversal of the Sequential element's child elements (lines 11 to 18). The duration of the Parallel element is determined by the duration of one of its child elements. This can be the child element with the maximum presentation time (lines 19 to 27), the element with the minimum presentation time, or a particular element i (not shown in Listing 6.1). With traversal of the leaf nodes, i. e., the media elements, for each medium element its position on the global timeline is captured (not shown in the listing).

```
function CalculateGlobalTimeline( node, time ) returns time
inputs: node, the current node in the multimedia document tree
        time, the current value of the global timeline

type <- GetNodeType( node )
case type of
  Medium, TemporalSelector, Delay:
    time = time + GetDuration( node )
    return time
  Sequential:
    subnodes <- Subnodes( node )
    loop do
      subnode <- Remove-Front( subnodes )
      if subnodes is empty then break
      time <- CalculateGlobalTimeline( subnode, time )
    end loop
    return time
  Parallel:
    startTime = time
    subnodes <- Subnodes( node )
    loop do
      subnode <- Remove-Front( subnodes )
      if subnodes is empty then break
      endTime <- CalculateGlobalTimeline( subnode, startTime )
      time <- Maximum( time, endTime )
    end loop
    return time
end case
```

Listing 6.1: Algorithm to calculate the global timeline from the local timelines

6.4.2 From Relative Positioning to Absolute Positioning

For presentation formats that allow the relative positioning of media elements, as it is supported by SVG and XMT-Ω for example, the internal multimedia content model represents already the spatial model of the target format. Here, the projectors of the internal multimedia content representation model are directly converted to the syntax of the presentation format. If a target format provides a spatial model with absolute positioning, the projectors are recalculated to absolute positions in space by applying the algorithm in Listing 6.2. The algorithm starts with the root element of the document tree. The initial reference point to calculate the absolute positions of the visual media elements constitutes the origin of the global coordinate system, i. e., the coordinate system of the entire multimedia presentation. The corresponding initial call of the algorithm is CalculateAbsolutePositions(rootnode, 0, 0). As the transformation from local timelines to a global timeline, the algorithm traverses the document tree from left to right and depth first. For each visual element the projectors determine the position in space. Each time when a SpatialProjector is attached to a node (lines 6 and 7) the projector's x and y value are added to the accumulated current absolute position (lines 8 and 9). The new absolute position (lines 10 and 11) are the reference values for the further traversal of the nodes children (lines 14 to 18). For each visual medium element its absolute position in space is stored during traversal. So, for each visual medium element the scope is mapped

from its prior local coordinate system to the global coordinate system of the entire multimedia presentation.

```
function CalculateAbsolutePositions( node, x, y )
inputs: node, the current node in the multimedia document tree
        x, position of the current node on the x-axis
        y, position of the current node on the y-axis

if node has SpatialProjector then
  spatial-p <- GetSpatialProjector( node )
  x <- x + GetXPosition( spatial-p )
  y <- y + GetYPosition( spatial-p )
  SetXPosition( spatial-p, x )
  SetYPosition( spatial-p, y )
end if
subnodes <- Subnodes( node )
loop do
  if subnodes is empty then break
  subnode <- Remove-Front( subnodes )
  CalculateAbsolutePositions( subnode, x, y )
end loop
```

Listing 6.2: Algorithm to calculate absolute positions from local coordinate systems

6.4.3 Transforming the Interaction Model

If the targeted presentation format supports external and internal links, both can be mapped by using the respective syntactic elements of the target format. Ideally the transformation process would merely translate every navigational link node in the multimedia document tree into the syntax of the targeted presentation format. Whereas the transformation of the temporal and spatial model into the target formats listed in Table 6.1 results in an accurate rendering of the presentation by available players this does not hold for navigational links. Not all multimedia players support internal links even though the specification of the presentation format defines them, e. g., the Internet Explorer in conjunction with Adobe's SVG Viewer 6.0 Pre-Release plug-in [Ado06a]. Also referring to a distinct medium element of an external presentation is not supported by all players, e. g., the RealPlayer 10.5 presenting SMIL [Rea06]. Consequently, for practical reasons the transformers have to be not only format specific, but also player specific: If internal links are not the supported, the document tree is split up into separate document trees for the target of each internal link. The internal links are thereby converted to respective external links referring to newly created external multimedia documents (each representing an internal link's target). If the player does not support linking to a particular fragment of an external presentation, there are two options: Either the external link is changed to refer just to the entire external presentation or the fragment of the external presentation needs to be extracted and presented as a single external document.

6.4.4 Mapping of Abstract Model to the Syntax of the Presentation Format

Besides the transformation of the temporal, spatial, and interaction model, the internal multimedia content representation model is mapped to the concrete syntax of the targeted format. Our internal representation model consists only of a small set of domain independent and therefore universally applicable multimedia composition elements. For each of these composition elements there exist adequate syntactic elements in the targeted presentation formats such as SMIL and SVG. Consequently, a mapping of the abstract composition elements to the syntax of a concrete presentation format by means of some distinct sets of transformation rules and mapping rules is possible. Figure 6.24 gives an example how the syntactic transformation is actually realized. It depicts a simple slideshow presentation represented in our abstract multimedia content representation model. We show how this example is transformed in the presentation formats SMIL 2.0, SMIL 2.0 BLP, and SVG 1.2. For SMIL 2.0 the abstract document tree reflects already the temporal and spatial model of this presentation format. In contrary SMIL 2.0 BLP does not support relative positioning and SVG 1.2 does not support local timelines. These target formats cover the different presentation characteristics as identified by our analysis and listed in Table 6.1 on page 102. In Figure 6.24, the encircled numbers at the edges of the graph indicate the order in which the nodes are transformed to the target format. The transformation results for SMIL 2.0, SMIL 2.0 BLP, and SVG 1.2 are depicted in Listing 6.3 to 6.5. The numbers at the beginning of each line indicate which lines are created by which step of the transformation process and allow to keep track of the transformation process.

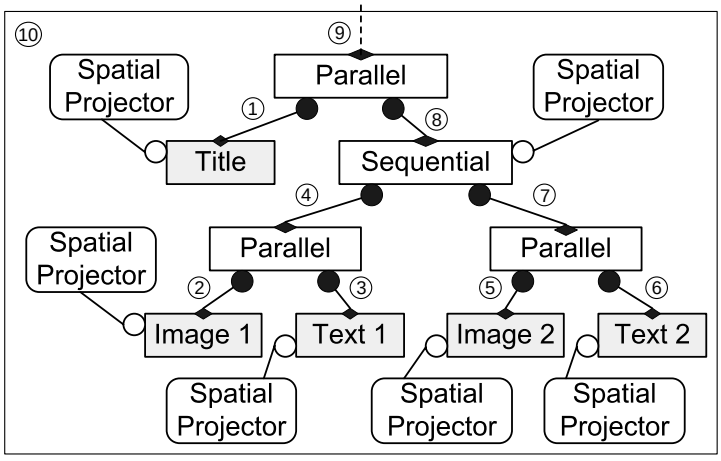


Figure 6.24: Slideshow in abstract model that is transformed to different presentation formats

The example of the transformation step shows only the transformation of a limited set of the available multimedia composition functionality defined for the internal multimedia content representation model. For example, it does not show the

transformation of all basic composition operators, e. g., the interactive operator Link. In the case of the Link operator, the corresponding syntactic element of the targeted presentation formats are in the case of SMIL, SVG, and (X)HTML the <a> tag. In addition, the example does also not show the transformation to all the different presentation formats.

The transformation process starts at the root element of the multimedia document tree. It traverses the document tree from left to right in depth first order. Each time the algorithm reaches a node which has only media elements as children, these media elements and their associated projectors are transformed into the corresponding syntactic elements of the target format.[2] The first time this is the case is for "Image 1" and "Text 1". The Image element "Image 1" is syntactically represented in SMIL by using the <img> tag and in SVG by <image> (1). For the Text medium "Text 1" a particular instance of the <text> element is used (2). At the same time the syntactic elements that determine the spatial layout of the presentation are created. Therefor, the SpatialProjectors that are attached to the media elements are transformed to respective <region> elements in SMIL and <g> tags in SVG. Then, the composition element Parallel itself is transformed (3). In the case of SMIL the element <par> is used, which has the same semantics as the abstract composition element Parallel in our representation model. Since SVG does not provide an appropriate syntactic element for Parallel, the media elements "Image 1" and "Text 1" are just placed at the same point in time on the global timeline to simulate its functionality. These global points in time are calculated by the algorithm presented in Section 6.4.1. The transformation process continues with the second child of the Sequential element and transforms it analogously to the first one (4 to 6). Then the composition element Sequential is mapped to the target format: Here, the <seq> tag is used in the case of SMIL. For SVG the media elements of the two slides are placed on the global timeline in a row by setting the values of the begin attributes to the corresponding global points in time. As the media elements, also the Sequential element has a SpatialProjector attached. This projector affects all the projectors of the media elements in the sub-tree. To realize such a nested spatial layout in the target formats, SMIL 2.0 and SVG 1.2 allow nesting the respective <region> and <g> elements. Therefore, with the SpatialProjector attached to the Sequential element a sub-region is created that shifts the scope of the coordinate system to a particular point in space to which the visual media elements of the sub-tree then refer to (7). This is syntactically realized in the target formats by embracing the <region> respectively <g> tags of the media elements with a <region> respectively <g> tag of the projector attached to the Sequential element. In the case of SMIL 2.0 BLP where only absolute positioning of media elements is supported, the <region> tags cannot be nested. Therefore, the absolute positions calculated by the algorithm presented in Section 6.4.2 are used for the region's attributes left and top. After the Sequential element is successfully transformed, the transformation process continues with mapping the medium element "Title" and the root element Parallel to the

[2] In fact, these are two sequential steps if the targeted presentation format defines an explicit area for determining the layout, e. g., within the <head> tag in SMIL and XMT-Ω. However, we consider them here as being processed at the same time for reasons of simplicity. In case that there is no explicit header tag, such as with the SVG family, the transformation of the content and the layout is conducted in parallel.

target format (8 and 9). Finally, the transformation process finishes with creating the syntactic frame of the entire presentation (10).

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head title="Slideshow — (C) 2006 Ansgar Scherp, OFFIS, Oldenburg, Germany">
    <layout type="text/smil-basic-layout">
      <root-layout width="720" height="597"/>
      <region id="title" left="0" top="0" width="125" height="37"/>
      <region id="slide_region" left="0" top="37" width="720" height="560">
        <region id="slide" left="0" top="0" width="720" height="540"/>
        <region id="text" left="0" top="540" width="720" height="20"/>
      </region>
    </layout>
  </head>
  <body>
    <par>
      <img region="title" src="title.png" begin="0s" dur="20s"/>
      <seq>
        <par>
          <img region="slide" src="slide1.jpg" begin="0s" dur="10s"/>
          <text region="text" src="text1.txt" begin="0s" dur="10s"/>
        </par>
        <par>
          <img region="slide" src="slide2.jpg" begin="0s" dur="10s"/>
          <text region="text" src="text2.txt" begin="0s" dur="10s"/>
        </par>
      </seq>
    </par>
  </body>
</smil>
```

Listing 6.3: The slideshow transformed to SMIL 2.0

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head title="Slideshow — (C) 2006 Ansgar Scherp, OFFIS, Oldenburg, Germany">
   <layout type="text/smil-basic-layout">
    <root-layout width="720" height="597"/>
    <region id="title" left="0" top="0" width="125" height="37"/>
    <region id="slide" left="0" top="37" width="720" height="540"/>
    <region id="text" left="0" top="577" width="720" height="20"/>
   </layout>
  </head>
  <body>
    [...]
  </body>
</smil>
```

Listing 6.4: The slideshow transformed to SMIL 2.0 BLP

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<svg xmlns="http://www.w3.org/2000/svg" width="720" height="597" version="1.2">
  <title>Slideshow — (C) 2006 Ansgar Scherp, OFFIS, Oldenburg, Germany</title>
  <image x="0" y="0" width="125" height="37" xlink:href="title.png"
      style="visibility:hidden;">
    <set attributeName="visibility" attributeType="CSS" to="visible" begin="0s"
```

```
        dur="20s" fill="remove"/>
    </image>
    <g id="slide_region" transform="translate(0,37)">
      <g transform="translate(0,0)">
        <image x="0" y="0" width="720" height="540" xlink:href="slide1.jpg"
            style="visibility:hidden;">
          <set attributeName="visibility" attributeType="CSS" to="visible"
              begin="0s" dur="10s" fill="remove"/>
        </image>
      </g>
      <g transform="translate(0,540)">
        <text x="0" y="10" style="visibility:hidden;fontsize:24;">
          <set attributeName="visibility" attributeType="CSS" to="visible"
              begin="0s" dur="10s" fill="remove"/>
          This is the description of slide one.
        </text>
      </g>
      <g transform="translate(0,0)">
        <image x="0" y="0" width="720" height="540" xlink:href="slide2.jpg"
            style="visibility:hidden;">
          <set attributeName="visibility" attributeType="CSS" to="visible"
              begin="10s" dur="10s" fill="remove"/>
        </image>
      </g>
      <g transform="translate(0,540)">
        <text x="0" y="10" style="visibility:hidden;fontsize:24;">
          <set attributeName="visibility" attributeType="CSS" to="visible"
              begin="10s" dur="10s" fill="remove"/>
          This is the description of slide two.
        </text>
      </g>
    </g>
  </svg>
```

Listing 6.5: The slideshow transformed to SVG 1.2

This small example of a slideshow presentation illustrates already how the transformation of the abstract document tree to the syntax of a concrete presentation format is realized. Since arbitrary multimedia content can be created with our internal document model, the transformation process is very flexible and also applicable for any application area.

In the following sections, we present a semi-formal description of the transformation rules that are used for mapping the basic composition elements exploited by the slideshow example into the syntax and features of the considered presentation formats. We start in Section 6.4.5 with the presentation of the transformation rules to the SMIL 2.0 format. Where necessary, we point out the particularities of the SMIL 2.0 BLP format as well as the RealPlayer [Rea06] specific characteristics. In Section 6.4.6, the transformation rules to SVG are presented. Besides the transformation rules for SMIL and SVG, we present in Section 6.4.7 the rules for transforming the internal representation model to the Flash Markup Language. The Flash Markup Language is a textual representation of the binary Flash format [Mac04]. The transformation rules presented in the following sections comprise selected multimedia composition elements of the internal multimedia content representation model.

6.4.5 Transformation Rules to SMIL

The transformation rules applied for mapping the internal multimedia content representation model to the syntax of the multimedia presentation format SMIL are presented in the following tables. The mapping of the media elements Image, Text, Video, and Audio are presented in Table 6.2. In addition to the media elements defined in the internal representation model, also the mapping of the composition operator TemporalSelector is presented. Where appropriate, also the application of the SpatialProjector and/or AcousticProjector is shown in the table.

As shown in Table 6.2, the W3C standard SMIL does not allow for formatting text elements. However, only unformatted plain text is supported. In the case that the text content is formatted in HTML-style, at best plain black text will be displayed by SMIL players which do not support for such formatted text [Pih03]. In worse case, the playback of a presentation including formatted text fails. Formatted text content can be displayed and is provided in SMIL, e. g., with the RealPlayer [Rea06]. However, the formatted text content rendered by the RealPlayer needs to be encoded in the proprietary RealText format [Rea01]. As Table 6.3 shows, RealText uses a specific <window> tag containing HTML-formatted text. The text's spatial extension is determined by the attributes width and height of the <window> tag.

The transformation of the temporal composition operators Sequential, Parallel, Delay, and Loop are shown in Table 6.4. In addition, the transformation of the composition operators for navigational interaction Jump and Link are presented. Finally, in Table 6.5 the transformation of the SpatialProjector and AcousticProjector to the SMIL 2.0 format are presented. The table also shows the particularities that have to be taken into account when transforming the projectors to the SMIL 2.0 BLP format.

In regard of the latest W3C recommendation of SMIL 2.1 [SMI05], there are some changes to the SMIL 2.0 standard. The SMIL 2.1 recommendation defines for example new transition effects and some other fancy features. However, there is nothing crucial new that needs additional consideration of SMIL 2.1 for the transformation rules. It is interesting to observe that with SMIL 2.1 two new profiles are defined. The first profile called SMIL 2.1 Mobile Profile is aimed for standard mobile devices and the second for extended mobile devices is named SMIL 2.1 Extended Mobile Profile. The main difference between the two profiles from the transformation rules point of view is that the SMIL 2.1 Mobile Profile only supports a flat spatial layout like the SMIL 2.0 Basic Language Profile does. With the SMIL 2.1 Extended Mobile Profile nesting of <region> tags like with the SMIL 2.0 full standard is supported.

6.4.6 Transformation Rules to SVG

The transformation rules applied for mapping the internal multimedia content representation model to the syntax of SVG are depicted in the following tables. The transformation of the representation model's Image and Text element are shown in Table 6.6. The mapping of the media elements Video and Audio are presented in Table 6.7. As shown in the table, there is no support by SVG for starting the playback at a specific point in time within a video element or audio element, like it is supported with, e. g., the clipBegin attribute in SMIL.

Basic composition elements	SMIL tag and attributes	Values of the attributes
Image	**<img>**	
- id	- id	Image element ID
- uri	- src	URI of the image element
- description	- alt	Image element description (optional)
- width, height	n/a	See transformation rule for SpatialProjector
TemporalSelector		
- start	n/a	Attribute is not needed for discrete media elements
- duration	- dur	Duration of image element playback
SpatialProjector		
- id	- region	SpatialProjector's region ID
Text	**<text>**	
- id	- id	Text element ID
- uri	- src	'data:,' + unformatted text content
- description	- alt	Text element description (optional)
- width, height	n/a	See transformation rule for SpatialProjector
TemporalSelector		
- start	n/a	Attribute is not needed for discrete media elements
- duration	- dur	Duration of text element playback
SpatialProjector		
- id	- region	SpatialProjector's region ID
Video	**<video>**	
- id	- id	Video element ID
- uri	- src	URI of the video element
- description	- alt	Video element description (optional)
- width, height	n/a	See transformation rule for SpatialProjector
TemporalSelector		
- start	- clipBegin	Start within the video element
- duration	- dur	Duration of the video element playback
SpatialProjector		
- id	- region	SpatialProjector's region ID
AcousticProjector		
- id	- region	AcousticProjector's region ID
Audio	**<audio>**	
- id	- id	Audio medium ID
- uri	- src	URI of the audio medium
- description	- alt	Audio medium description (optional)
TemporalSelector		
- start	- clipBegin	Start within the audio clip
- duration	- dur	Duration of the audio element playback
AcousticProjector		
- id	- region	AcousticProjector's region ID

Table 6.2: Transformation rules for mapping the media elements to SMIL

The transformation of the temporal composition operators to SVG is shown in Table 6.8. In addition, the table depicts the transformation rules to support for navigational interaction within SVG. In Table 6.9, an example of a player specific transformation rule for SVG is presented. It shows the mapping of the Image element to the TinyLine SVG player [Gir06]. The TinyLine SVG player is targeted at mobile devices with limited resources and thus aims at providing support for SVG Tiny [AAA+04a]. Some tags specified in the respective profile of the SVG standard

Basic composition elements	SMIL tag and attributes	Values of the attributes
Text	**<text>**	
- id	- id	Text element ID
- uri	- src	'**<window** width="<*width*>" height="<*height*>" **>**' + HTML-formatted text + '**</window>**'
- description	- alt	Text element description (optional)
- width, height	n/a	Is used within **<window>** tag (see also transformation rule for SpatialProjector)
TemporalSelector		
- start	n/a	Attribute is not needed for discrete media elements
- duration	- dur	Duration of text element playback
SpatialProjector		
- id	- region	SpatialProjector region ID

Table 6.3: Transformation rule for mapping the Text element to RealText

are not supported by the TinyLine SVG player, e. g., the <set> tag. Thus, we had to replace the <set> tag used for the SVG transformation rules by the more complicated solution of four <animate> tags.

6.4.7 Transformation Rules to Flash

Adobe's Flash format is currently the most important multimedia presentation format on the Internet. Although, it is a proprietary format, today the Flash player [Ado06c] is available on almost every computer connected to the Internet [Mac03, Ado06d]. The Flash player software is available for a huge variety of platforms, including Desktop PCs, PDAs, and cell phones, as well as for many operating systems. Consequently, the developed MM4U framework shall also be capable of creating personalized Flash presentations.

Flash is a simple-structured binary multimedia presentation format [Mac04]. Being a binary format is the most obvious difference between the Flash format and the other XML-based presentation formats such as SMIL and SVG. For creating personalized multimedia presentations in Flash, we could directly generate the binary Flash content. However, debugging would be difficult as only the binary presentation would be available for troubleshooting. Consequently, we divide the creation of Flash presentations in two steps. First, we generate a XML-based representation of the binary Flash content, before the actual binary Flash file is created. For it, we define a XML-based representation of the binary Flash format called the Flash Markup Language (FML). The FML is designed to be easily transformed to the binary format. This is achieved by targeting at a low-level and closely to the binary format oriented XML-representation.

In the following Section 6.4.7.1, a brief introduction to the binary Flash format is presented. This is necessary in order to define and introduce the FML in Section 6.4.7.2. The transformation rules to map the internal multimedia content representation model to the FML are described in Section 6.4.7.3. In Section 6.4.7.4,

Basic composition operator	SMIL tag, attributes, and elements	Values of the attributes and elements
Sequential - id	**<seq>** - id Child elements	 Sequential operator ID Arbitrary number of media elements or other presentation fragments
Parallel - id - finish	**<par>** - id - endsync Child elements	 Parallel operator ID 'first', 'last' (default), or particular child element ID Arbitrary number of media elements or other presentation fragments
Delay - id - delay	**<seq>** - id - dur	 Delay operator ID Delay time
Link - id - target - mode	**<a>** - id - href - show Child element(s)	 Link operator ID Link to internal or external resource Open in new window or replace current presentation: 'new' or 'replace' One arbitrary visual medium element or presentation fragment

Table 6.4: Transformation rules for mapping the composition operators to SMIL

Projector	SMIL tag, attributes, and elements	Values of the attributes
SpatialProjector - id - x, y - width, height - priority - unit	**<region>** - id - left, top - width, height - z-index n/a Child elements	 Spatial projector ID x-position, y-position Width and height of projector Z-order priority Selected unit, e. g., 'px' for pixel; added to left, top, width, and height attributes Other **<region>** tags (SMIL 2.0) or nothing (SMIL 2.0 Basic Language Profile)
AcousticProjector - id - volume - balance	**<region>** - id - soundLevel n/a Child elements	 Acoustic projector ID Volume of audio element or video element Other **<region>** tags (SMIL 2.0) or nothing (SMIL 2.0 Basic Language Profile)

Table 6.5: Transformation rules for mapping the layout elements to SMIL

the relation of the FML to other approaches abstracting from the binary Flash format is presented.

6.4.7.1 Brief Introduction of the Binary Flash Format

Before we come to the presentation of our Flash Markup Language in the next section, it is crucial to gain some knowledge about the fundamental structure of the binary Flash format [Mac04] first. Besides being a binary format, also the internal structure of a Flash presentation is very different to the other formats, as it bases on a sequence of so-called *frames* for modeling the temporal course of the multimedia presentation. Consequently, with Flash a medium element cannot be associated directly to a specific time interval like, e. g., provided with the begin attribute and dur attribute of SVG for determining when the playback of a medium element starts and for specifying its presentation duration.

A Flash file starts with a header, containing information about the presentation's frame rate, the total number of frames, and the file size. Although Flash is a binary format, its internal structure is tag-based. Subsequent to the header, the remainder of a binary Flash file consists of a sequence of binary Flash-tags. A Flash file is closed by a so-called end-tag. Each binary-tag defines a data structure, containing the type, length, and unique identifier of the tag. In addition, it also contains tag-specific attributes [Mac04]. For the spatial positioning of visual media elements, Flash uses an absolute positioning model (see Section 3.1.2). These absolute positions in space must be defined in an own unit called TWIP (short for "TWentIeth of a Point") and is a $1/20$ of a logical pixel of the display. As stated above, the temporal course of a Flash presentation is modeled by using the concept of frames. In Flash, each frame has the same duration, i. e., the duration of a frame is constant within the presentation.[3] It is possible to add frames at arbitrary positions in the Flash presentation. Thereby, all the following frames are shifted n frames to the back. Consequently, a time model of local timelines is provided.

For realizing interaction of the presentation with the user, Flash provides extensive scripting functionality with its build-in script language ActionScript. The navigational interactions required for the MM4U framework are provided by "invisible buttons" laying over the visual media elements. Once a user clicks on such a button, the corresponding ActionScript-code is executed and the navigational link is followed. In addition to the active navigational interaction, also non-active navigational jumps are provided by Flash that are executed without the user clicking on it.

The binary tags of the Flash format can be divided into two categories (the only exception is the tag for setting the background color). The first category comprises the definition tags. With these tags, the media elements used within the Flash presentation are defined. The second category is the control tags. These determine when and where the media elements shall be presented during the playback of the presentation, i. e., they define the temporal course and spatial layout of the presentation. In addition, they determine the interaction possibilities of the user with the presentation. Before using the control tags, the corresponding media elements must be defined within the presentation header by using the definition tags.

[3] However, this constant is adjustable by the frame rate determined in the header.

For managing the media elements used within a Flash presentation and for specifying the temporal course of presenting these media elements, the so-called display list is provided by Flash. The display list is an internal data structure comprising the tags of all media elements that are presented with the next frame of a Flash presentation. Thus, adding and removing media elements from the display list changes the playback of the Flash presentation over time. When a medium element shall be presented with the next frame and it is not already registered in the display list, the corresponding tag is added to the list. For stopping the playback of a specific medium element with the next frame, its tag is removed from the display list. If a medium element shall be presented over multiple frames, the media element's tag does not need to be registered for each new frame. Once it is registered in the display list it is left there over the specified period of time. Only with the last frame of the medium element's playback, the corresponding tag needs to be removed from the display list.

6.4.7.2 Definition of the Flash Markup Language (FML)

The Flash Markup Language (FML) defines a XML-based representation of central parts of the binary Flash format [Mac04]. The FML provides support for image, text, video, and audio elements. However, it does not provide for modeling geometric shapes such as circles and rectangles. The binary Flash format provides with its scripting language ActionScript support for navigational interaction. The FML employs this scripting functionality by defining tags, e. g., for opening an external multimedia presentation determined by a URL and jumping to a specific position within the presentation (i. e., jumping to a specific frame). To keep the FML as closely as possible to the internal structure of the binary Flash format, the temporal course of the presentation is defined in the FML on the basis of Flash frames and not by points in time of a global timeline as to be found with the multimedia presentation format SVG.

However, we also conducted some simplifications of the binary Flash format where necessary. For example, for creating an image element in Flash, two tags of the binary Flash format are required (these are DefineJPEG2 or DefineBitsLossless2 and DefineShape2). In the FML, this is shortened by introducing the tag <DefineImage>. In addition, the basic unit used for determining spatial positions in the FML is pixel, although the binary Flash format requires TWIP for defining the spatial positions of the visual media elements. However, despite of the simplifications conducted, the FML still carries all necessary information required by the binary Flash format.

The structure of an FML-document consists of the root element <FlashMovie> and three main nodes. These three main nodes are, corresponding to the binary Flash format, the <Header>, <Define>, and <Body> tag and are described in the following.

1. <Header>: The header node comprises the following tags:
   - <FrameSize>: Determines the width and height of the Flash presentation.
   - <BackgroundColor>: Defines the background color of the presentation in RGB-values.

2. <Define>: Within this node, the media elements that are used for the Flash presentation are defined. For it, the FML provides the following tags:

   - <DefineImage>: This tag defines an image element. It possesses the two identifier attributes imageID and characterID, as well as attributes for determining the width and height, and the resource location of the image element (in form of a URL). The two identifiers are required since the binary Flash format internally needs two binary tags to display an image element. One for the image element's raw media data and the second one to draw a rectangle of a corresponding size which is filled with the image element's media data. The FML-tag <DefineImage> can be used for both lossy and lossless image formats such as JPG as well as GIF and PNG.
   - <DefineText>: With the <DefineText> tag, an arbitrary XHTML-formatted text element can be defined in FML. The formatted text is embedded in the <DefineText> tag. The tag also provides attributes for determining the width and height of the text element and has a unique identifier.
   - <DefineVideo>: This tag is used to define a video element within a multimedia presentation. The <DefineVideo> tag possesses a unique identifier and provides attributes for determining the width and height of the video element. The video elements resource can be specified by a URL.
   - <DefineSound>: Defines an audio element, e. g., in the MP3 format. The <DefineSound> tag has a unique identifier and specifies the audio element resource in form of a URL.

3. <Body>: The third main node of an FML-document is the <Body> tag. Within this tag, the temporal course, spatial layout, and interaction possibilities of the multimedia presentation are specified. For determining the temporal course, the media elements defined in the <Define> section, are arranged on the global timeline of the Flash presentation, i. e., they are arranged on the sequence of the single Flash frames. For each medium element it is determined at which frame the playback of the medium element starts and at which frame the playback of the medium element stopps. For defining a frame in the FML the <ShowFrame> tag is provided. This tag is described in the following.

   - <ShowFrame>: The <ShowFrame> tag defines a new frame within a Flash presentation. Consequently, a sequence of <ShowFrame> tags determines a sequence of frames. Within a <ShowFrame> tag, an arbitrary number of tags can be embedded for adding and removing distinct media element from the presentation, i. e., for manipulating the display list of the Flash presentation (see Section 6.4.7.1). These tags include defining navigation interaction in the presentation. The <ShowFrame> tag provides two attributes, which are both optional. With the attribute name a label or anchor can be defined for a <ShowFrame> tag. This is required for realizing navigation within the presentation. Thus, the <ShowFrame> tag's label needs to be unique. The attribute duration determines how many Flash frames shall be presented until the next <ShowFrame> tag in the FML is processed. Thus, the duration attribute determines the number of frames following the current frame in which the display is not manipulated. This avoids the definition of a distinct <ShowFrame> tag for each Flash frame

and allows for easily presenting a determined sequence of equal Flash frames. Within a <ShowFrame> tag, the following control tags can be embedded for manipulating the display list of the Flash presentation.

- <PlaceObject>: This tag is used to add a visual medium element to the display list. Which medium element is added is determined by the media element's unique identifier specified in the <Define> section. The <PlaceObject> tag refers to the medium element's identifier by the attribute characterIDRef. Further attributes of the <PlaceObject> tag are the depth attribute determining the z-order value and the attributes x and y specifying the spatial position of the visual medium element in form of an absolute x,y-coordinate. In contrast to the other presentation formats such as SMIL and SVG, the z-order value must be unique in Flash as it is used internally by the Flash player for stopping the playback of media elements in the presentation.
  The <PlaceObject> tag provides two optional attributes target and mode. If the target attribute is defined, the referred visual medium element is click-able, i. e., if the user clicks on the image element the presentation jumps to the defined target. The target can be either an external multimedia presentation or a specific point in time (i. e., <ShowFrame> tag) within the presentation. The attribute mode determines whether the presentation is opened in a new window or if it is presented in the same window of the current presentation.
  To provide for some specific parts of the visual medium element to be click-able only, an arbitrary number of <Hotspot> tags can be embedded in the <PlaceObject> tag. Each <Hotspot> tag defines an rectangle area within the visual medium element that is click-able. This rectangle area is defined by the attributes x and y for specifying the area's upper left corner and the attributes width and height determining the spatial extension of the rectangle. In addition, each <Hotspot> tag has a target and mode attribute as defined above.
  In the case that a video element is referred by the <PlaceObject> tag, also the optional attribute volume is provided. This attribute allows for determining the volume of the video element's audio track. In addition, also the start within the video element can be determined by the optinal attribute clipBegin.

- <RemoveObject>: The <RemoveObject> tag removes a medium element from the display list with the next Flash frame. The medium element to be removed is identified by its unique z-order value (see <PlaceObject> tag). Like the <PlaceObject> tag, the <RemoveObject> tag can only be applied for visual media elements.

- <StartSound>: This tag starts an audio element, determined by its unique identifier. The optional attributes volume and balance determine the volume and balance of the audio element. With the optional attribute clipBegin, the start within the audio element can be determined. The <StartSound> tag can be applied for audio elements only.

- <StopSound>: This tag is applied to stop the playback of an audio element. Which audio element shall be stopped is determined by the audio element's identifier.
- <JumpTo>: With this tag, external and internal navigation can be realized that does not require user interaction. The attribute target determines, whether an external or internal target resource is addressed. The resource is specified in form of a URL. An external resource can be either a multimedia document, such as a SVG or SMIL presentation, a standard-HTML web page, or another (possibly dynamically generated) Flash presentation. For an internal resource, the label name of the corresponding <ShowFrame> tag is referred. The optional mode attribute determines whether the targeted presentation shall be opened in a new window or presented in the same window of the current Flash presentation.
- <DefineLoop> and <LoopTo>: The semantics of a loop is related to the semantics of the <JumpTo> tag. However, it provides a counter that is decreased each time the loop is executed. This counter is defined in the FML by the <DefineLoop> tag, which is determining a placeholder in the binary Flash format for the loop counter. It has an attribute loopName determining the unique name of the loop. The binary Flash format requires that this counter is defined in a frame prior to that frame where the actual loop starts. This is ensured by setting the <DefineLoop> tag in the <Define> section of our FML. For defining the start of a loop, i. e., for determining the frame where a loop begins, the attribute name is added to the corresponding <ShowFrame> tag. The value of the name attribute is the loop's unique counter name as defined in the <DefineLoop> tag. The end of a loop is determined by the <LoopTo> tag. Within this tag, the number of repeats of the loop is determined by the repeat attribute and thus holds the initial value of the loop's counter. Each time the <LoopTo> tag is executed, the counter is decreased by one until the loop finishes. The target attribute holds the name of the <ShowFrame> tag, where the start of the loop is defined.

Listing 6.6 shows an example of a multimedia presentation defined in the FML. The presentation is the slideshow example presented in Section 6.4.4. For a complete DTD of our FML we refer the reader to Appendix D.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FlashMovie SYSTEM "fml.dtd">
<FlashMovie>
 <Header flashVersion="7" frameRate="12">
   <FrameSize width="720" height="597"/>
   <BackgroundColor green="FF" red="FF" blue="FF"/>
 </Header>
 <Define>
   <DefineImage imageID="2" characterID="3" width="125" height="37"
     src="title.png"/>
   <DefineImage imageID="4" characterID="5" width="720" height="540"
     src="slide1.jpg"/>
   <DefineText characterID="6" width="720" height="20">
```

```
      This is the description of slide one.
    </DefineText>
    <DefineImage imageID="7" characterID="8" width="720" height="540"
      src="slide2.jpg"/>
    <DefineText characterID="9" width="720" height="20">
      This is the description of slide two.
    </DefineText>
  </Define>
  <Body>
    <ShowFrame duration="120">
      <PlaceObject characterIDRef="3" depth="5" x="0" y="0"/>
      <PlaceObject characterIDRef="5" depth="1" x="0" y="0"/>
      <PlaceObject characterIDRef="6" depth="2" x="0" y="540"/>
    </ShowFrame>
    <ShowFrame duration="120">
      <RemoveObject depth="1"/>
      <RemoveObject depth="2"/>
      <PlaceObject characterIDRef="8" depth="3" x="0" y="0"/>
      <PlaceObject characterIDRef="9" depth="4" x="0" y="540"/>
    </ShowFrame>
    <ShowFrame>
      <RemoveObject depth="5"/>
      <RemoveObject depth="3"/>
      <RemoveObject depth="4"/>
    </ShowFrame>
  </Body>
</FlashMovie>
```

Listing 6.6: The slideshow transformed to Flash Markup Language

6.4.7.3 Transformation from the Internal Representation Model to FML

In the previous sections, we introduced the fundamental structure and concepts of the binary Flash format. We have defined a XML-based markup language for Flash with the FML. The FML is defined as closely as possible to the binary format to allow for an easy transformation to the binary Flash format. In this section, we introduce the rules applied for mapping the composition elements of the internal multimedia content representation model to the FML. In Table 6.10, the mapping of the media elements Image and Text to the FML is depicted. The mapping of the media elements Video and Audio is shown in Table 6.11

The Flash format provides for a global timeline determined by the sequence of its frames. The basic multimedia composition operators Sequential, Parallel, and Delay are internally realized by the algorithm for calculating the global timeline (see Section 6.4.1). This allows for appropriately applying the <PlaceObject> and <RemoveObject> as well as the <StartSound> and <StopSound> tags on the Flash frames of the presentation's temporal course. In addition, the navigational interaction is provided by the <PlaceObject> tag, which is part of the transformation rules for the media elements shown in Tables 6.10 and 6.11.

6.4.7.4 Relation of FML to Similar Approaches

Besides the FML, there exist also other approaches that provide for an abstraction from the binary Flash format. These related approaches are Flex [Ado06e] provided

by Adobe and the open source platform OpenLaszlo [Las06] originally developed by Laszlo Systems. The FML distinguishes from these approaches as both are targeted at providing a complete development environment for developing interactive web-based applications on the basis of Flash movies, e. g., an online web shop. In addition, both approaches define a programming language for abstracting from the binary Flash content. They are equipped with an appropriate compiler and runtime class library. Finally, both come with a sophisticated deployment mechanism targeted at well-equipped Internet servers.

In contrast to Flex and Open Laszlo, our FML is very slim. It only defines a straight-forward XML-based representation of the binary Flash format. A multimedia presentation defined in FML can be directly mapped to the binary Flash format. Employing our FML allows for generating Flash presentations also on limited devices such as PDAs, as it does not come with the burden of a sophisticated development environment and runtime library.

6.5 Summary

In this section, we have presented the concepts and data models underlying to the MM4U framework in detail. We have presented the modeling of media data and associated meta data. A unifying approach for user modeling has been introduced, which embraces the issues of both the area of classical user modeling as well as the area of context modeling. An abstract multimedia content representation model has been defined as well as an application-independent transformation algorithm. Transformation rules are introduced that allow for transforming the abstract representation model into the features and syntax of todays concrete presentation formats. This multimedia content representation model and application-independent transformation algorithms allow for a presentation independent generation and distribution of personalized multimedia content (see requirement for presentation independence in Section 5.2.2).

In the following, the actual development and implementation of the MM4U framework is described. For it, state-of-the-art software engineering technology shall be applied. Thus, we aim at employing component technology as the currently most promising approach for implementing reusable software architectures. Consequently, we target at developing a *component* framework for personalized multimedia applications. As described in Section 4.2.3, a proper process model and development method for component frameworks is still missing. To improve the development process of component frameworks, we developed the ProMoCF approach. This constitutes a lightweight process model and development method for component frameworks and is described in the following Section 7. The employment of the ProMoCF approach for the development of the MM4U component framework is presented in Section 8.

Basic composition elements	SVG tag, attributes and elements	Values of the attributes
Image	**<image>**	
- id	- id	Image element ID
- uri	- xlink:href	URI of the image element
- description	- alt	Image element description (optional)
- width, height	[- width, height]	Only used if no SpatialProjector is provided
SpatialProjector		
- x, y	- x, y	Image element's x,y-position
- width, height	- width, height	Width and height of image element
	Additional attribute:	
	- style	'visibility:hidden;' (Default is to hide the image element)
TemporalSelector	Embedded **<set>** tag:	Used to show and hide the image element:
- start	- begin	Global start time when the image element shall be presented (calculated by the global timeline algorithm)
- duration	- dur	Duration of image element playback
	Additional attributes:	
	- attributeName	'visibility' (Modify the attribute 'visibility')
	- attributeType	'CSS'
	- to	'visible' (Change the value of 'visibility' of attribute style to 'visible', i. e. make the image element visible at the point in time specified by the begin attribute for the duration determined by the dur attribute)
Text	**<text>**	
- id	- id	Text element ID
- uri	n/a	List of nested **<tspan>** tags for formatted text content
- description	- alt	Text element description (optional)
- width, height	n/a	Width and height is realized implizitly by **<tspan>** tags
SpatialProjector		
- x, y	- x, y	Text element's x,y-position
- width, height	n/a	Width and height is realized by **<tspan>** tags
	Additional attribute:	
	- style	Default text format: 'visibility:hidden;fill:#000000;font-size:14px; font-family:Times;font-weight:normal;text-decoration:none;'
	Embedded **<tspan>** tags:	Provides for HTML-like formatted text content (each row of the text content is realized as single **<tspan>** tag).
	- font-family	'font-family:' + *<font name>*, e.g., Times
	- style	Parameters of style attribute are: 'font-size:' + *<font size>*, 'font-weight:' + <'bold'|'italic'>, 'text-decoration:underline', 'fill:' + *<font color>*, e. g., #FF00FF
	- x	The **<tspan>** text content x-position
	- dy	Delta y-position to the previous **<tspan>** tag
TemporalSelector	Embedded **<set>** tag	Embedded **<set>** tag as with the image element above
- start, duration		

Table 6.6: Transformation rules for mapping the Image element and Text element to SVG

Basic composition elements	SVG tag and attributes	Values of the attributes
Video	**<video>**	
- id	- id	Video element ID
- uri	- xlink:href	URI of the image element
- description	- alt	Video element description (optional)
- width, height	[- width, height]	Only used if no SpatialProjector is provided
TemporalSelector		
- start	- begin	Global start time when the video element shall be presented (calculated by the global timeline algorithm). However, *clip-begin* is not supported by SVG.
- duration	- dur	Duration of video element playback
SpatialProjector		
- x, y, width, height	- x, y, width, height	Video element's x,y-position as well as width and height
AcousticProjector		
- volume	- volume	Volume of video element
- balance	n/a	
Audio	**<audio>**	
- id	- id	Audio element ID
- uri	- xlink:href	URI of the audio element
- description	- alt	Audio element description (optional)
TemporalSelector		
- start	- begin	Global start time when the audio element shall be presented (calculated by the global timeline algorithm). However, *clip-begin* is not supported by SVG.
- duration	- dur	Duration of audio element playback
AcousticProjector		
- volume	- audio-level	Volume of audio element
- balance	n/a	

Table 6.7: Transformation rules for mapping the Video element and Audio element to SVG

Basic composition elements	**SVG tag, attributes and elements**	**Values of the attributes**
Sequential - id **SpatialProjector** - x, y	**<g>** - id - transform Child elements	 Sequential operator ID 'translate(*<x−position>*, *<y−position>*)' with x-position = x and y-position = y This attribute is set only, if there is a SpatialProjector directly associated to the Sequential operator. All composition elements of the Sequential operator are embedded in the **<g>**-tag. The timing of these elements is realized by the global time model of SVG.
Parallel - id - finish **SpatialProjector** - x, y	**<g>** - id n/a - transform Child elements	 Parallel operator ID Is implicitly realized by the global time model of SVG. See Sequential operator above. See Sequential operator above.
Delay - id - delay	n/a	Is implicitly realized by the global time model of SVG.
Link - id - target - mode	**<a>** - id - xlink:href - xlink:show Child element(s)	 Link operator ID Link to internal or external resource Open in new window or replace current presenation: 'new' or 'replace' One arbitrary visual medium element or presentation fragment

Table 6.8: Transformation rules for mapping basic composition operators to SVG

Medium element	TinyLine SVG tag, attributes and elements	Values of the attributes
Image	**<image>**	
- id	- id	Image element ID
- uri	- xlink:href	URI of the image element
- width, height	n/a	
TemporalSelector		
- duration	n/a	
SpatialProjector		
- x	- x	'-173741' (Position used to hide image element)
- y	- y	'-173741' (Position used to hide image element)
	Four **<animate>** tags embedded as follows:	
	1. **<animate>** tag	
	- attributeName	'x'
	- attributeType	'XML'
	- from	x-position as provided by SpatialProjector
	- to	x-position as provided by SpatialProjector
	- begin	Start time when the image element shall be presented (calculated by the global timeline algorithm)
	- dur	'1ms'
	- fill	'freeze'
	2. **<animate>** tag	
	- attributeName	'y'
	- attributeType	'XML'
	- from	y-position as provided by SpatialProjector
	- to	y-position as provided by SpatialProjector
	- begin	Start time when the image element shall be presented (calculated by the global timeline algorithm)
	- dur	'1ms'
	- fill	'freeze'
	3. **<animate>** tag	
	- attributeName	'x'
	- attributeType	'XML'
	- from	'-173741' (Position used to hide image element)
	- to	'-173741' (Position used to hide image element)
	- begin	End time of the presentation of the image element (calculated by the global timeline algorithm using the presentation duration provided by TemporalSelector)
	- dur	'1ms'
	- fill	'freeze'
	4. **<animate>** tag	
	- attributeName	'y'
	- attributeType	'XML'
	- from	'-173741' (Position used to hide image element)
	- to	'-173741' (Position used to hide image element)
	- begin	End time of the presentation of the image element (calculated by the global timeline algorithm using the presentation duration provided by TemporalSelector)
	- dur	'1ms'
	- fill	'freeze'

Table 6.9: TinyLine SVG player specific transformation rule for mapping the Image element to SVG

Basic composition elements	FML tags and attributes	Values of the attributes
Image	**<DefineImage>**	
- id	- characterID	Character ID of the image element's shape
- uri	- src	URI of the image element
- description	n/a	
- width, height	- width, height	Width and height of the image element
	Additional attribute:	
	- imageID	ID of the image medium (internal ID required by Flash to fill the shape with the image medium)
SpatialProjector	**<PlaceObject>**	
- x, y	- x, y	Image element's x,y-position
- priority	- depth	(1) Unique z-order value of the image element
Link		
- id	n/a	
- target	- target	Link to internal or external resource (optional)
- mode	- mode	Open in new window or replace current presenation: 'new' or 'same' (optional)
	Additional attributes:	
	- characterIDRef	Reference to the character ID above
TemporalSelector		
- start	n/a	Attribute is not needed for discrete media elements
- duration	n/a	Implicitly realized by the global time model of Flash and using the **<RemoveObject>** tag
	<RemoveObject>	
	- depth	Image element's z-order value as defined above at (1)
Text	**<DefineText>**	
- id	- characterID	Character ID of the text element
- uri	n/a	Text element's content is embedded inline (see below)
- description	n/a	
- width, height	- width, height	Width and height of the text element
	Embedded element	The actual text content, formatted in HTML-style.
SpatialProjector	**<PlaceObject>**	
- x, y	- x, y	Text element's x,y-position
- priority	- depth	(2) Unique z-order value of the text element
Link		
- id	n/a	
- target	- target	Link to internal or external resource (optional)
- mode	- mode	Open in new window or replace current presenation: 'new' or 'same' (optional)
	Additional attributes:	
	- characterIDRef	Reference to the character ID above
TemporalSelector		
- start		
- duration	n/a	Attribute is not needed for discrete media elements
	n/a	Implicitly realized by the global time model of Flash and using the **<RemoveObject>** tag
	<RemoveObject>	
	- depth	Text element's z-order value as defined above at (2)

Table 6.10: Transformation rule for mapping the Image element and Text element to FML

Basic composition elements	FML tags and attributes	Values of the attributes
Video	**<DefineVideo>**	
- id	- characterID	Character ID of the video element
- uri	- src	URI of the video element
- description	n/a	
- width, height	- width, height	Width and height of the video element
SpatialProjector	**<PlaceObject>**	
- x, y	- x, y	Video element's x,y-position
- priority	- depth	(3) Unique z-order value of the video element
Link		
- id	n/a	Link to internal or external resource (optional)
- target	- target	Open in new window or replace current presenation:
- mode	- mode	'new' or 'same' (optional)
	Additional attributes:	
	- characterIDRef	Reference to the character ID above
AcousticProjector		
- volume	- volume	Volume of the audio playback of the video element
- balance	n/a	Attribute is not supported by Flash
TemporalSelector		
- start	clipBegin	Start within the video element
- duration	n/a	Implicitly realized by the global time model of Flash and using the **<RemoveObject>** tag
	<RemoveObject>	
	- depth	Video element's z-order value as defined above at (3)
Audio	**<DefineSound>**	
- id	- characterID	(4) Unique character ID of the audio element
- uri	- src	URI of the audio element
- description	n/a	
AcousticProjector	**<StartSound>**	
- volume	- volume	Volume of the playback of the audio element
- balance	- balance	Balance of the playback of the audio element
	Additional attributes:	
	- characterIDRef	Reference to the character ID above
TemporalSelector		
- start	clipBegin	Start within the audio element
- duration	n/a	Implicitly realized by the global time model of Flash
	<StopSound>	
	- characterIDRef	Character ID of the audio element as defined above (4)

Table 6.11: Transformation rule for mapping the Video element and Audio element to FML

7 ProMoCF—A Lightweight Process Model and Development Method for Component Frameworks

The development of the MM4U framework shall not be conducted *ad hoc* but base on applying a systematic process model accompanied by a proper development method for software frameworks. Hence, during the analysis and systematic categorization of related work in the field of personalizing multimedia content (see Sections 3.2 and 3.3) and the identification of the requirements to the MM4U framework (see Section 5.2), also a process model for software frameworks was selected. We decided in favor of the hot-spot-driven approach by Wolfgang Pree presented in Section 4.2.2, since it is a mature and proved process model and development method for the development of object-oriented frameworks. However, to provide support for the development of *component* frameworks, we need to modify Pree's original hot-spot-driven approach towards a lightweight process model and development method for component frameworks. In Section 7.1, we argue for the benefits of such a *lightweight* process model and development method for the development of software frameworks, before we present in Section 7.2 our modification of the Pree approach. This modification is called ProMoCF and extends the Pree approach by activities and methodical support for identifying the framework's components and specifying the framework's variability by group-hot-spot-cards.

7.1 Need for a Lightweight Process Model for Component Frameworks

As we considered in the beginning of Section 4, the development of software frameworks is an iterative process and should be conducted by applying an appropriate process model [CC02]. However, there are different kinds of process models, like heavyweight models, lightweight models, and agile models. From our experience, we argue that lightweight process models are most suitable for the development of software frameworks.

With agile process models, e. g., the very popular eXtreme Programming (XP) [Bec00], Cockburn's Crystal Family [Coc06], and De Luca's Feature Driven Development [Neb04], the software is developed and delivered in increments and anything beyond programming code is minimized [Som04]. Consequently, agile models do not provide a real system specification. For example, in the case of XP only rudimental use cases, so-called *stories*, are defined to describe the application's functionality.

This functionality, however, is only incrementally added to the application's design and only "the simplest thing that could possibly work" is implemented. Further flexibility requirements and the aspect of reuse of design are not considered. It is assumed that it is easier to change the original source code than adding flexibility in advance to provide support for reuse. This means that only as much flexibility is added to the application's design as it is needed to realize the actually required functionality. In other words, agile models do not target at developing reusable software architectures in the sense of the reuse characteristics of software frameworks. Although, agile process models like to exploit existing components to actually realize the required functionality and by this provide reuse of existing components, this does not help for developing reusable architectures, like component frameworks. As a consequence, we conclude that agile process models should not be used for developing frameworks.

Heavyweight process models, like the Rational Unified Process (RUP) [Kru00] and the Domain Specific Software Architecture (DSSA) engineering process [TTC95, Has02], provide a fully-fledged support for developing reusable software applications (in a specific domain). With the RUP, we find a process model that carries out the development of the application over several phases and iterations applying different workflows, e. g., for requirements analysis, design, implementation, test, and project management. These workflows can be tailored to the requirements of a specific domain. In the case of the DSSA engineering process, a domain model as well as a reference architecture for the considered domain is developed [TTC95] to support efficient application development in the domain. This is reflected by two concurrent processes for domain engineering and application engineering [Has02]. When adapting a heavyweight process model to the requirements of a specific domain, e. g., the RUP tailored and extended to the domain of virtual laboratories [Sch02], it provides for an efficient development of applications in that domain.

Applying a heavyweight process model for developing reusable software applications can be conducted in two different ways. First, the development of the reusable parts of the applications, e. g., a software framework, can be conducted in a proactive manner. With a proactive heavyweight process model, the software framework is developed prior to the concrete applications employing the framework. Second, the reusable parts can be developed in a reactive manner. In this case of a reactive heavyweight process model, first several concrete applications are developed in the domain. Only then in a second step, the abstract framework architecture is derived from the concrete applications. In principle, both variants are possible for the development of a software framework. However, when applying a proactive heavyweight process model for the development of a software framework, it has two significant drawbacks. First, it is difficult to develop a framework in advance without having already developed several applications in the domain. The framework developers might also not have the necessary experience in the considered domain to actually develop a framework for it. Second, a proactive process model requires a high initial investment in the development of the software framework. With developing the framework first, it takes a long time until the initial development effort for the framework yields in a return of investment by, e. g., selling concrete applications developed on basis of this framework. Thus, applying a proactive heavyweight process model for the development of a software framework is involved with a high initial investment and financial risk if the concrete applications miss the expected results.

Applying a reactive heavyweight process model for evolving frameworks would be a much more appropriate approach here, as first several applications are developed in the domain. In a second step, the desired framework architecture would be abstracted from these applications. However, creating a reactive process model for a specific domain requires high effort and expertise, which is not reasonable in all cases. Thus, it is debatable whether such a process model should be created for the specific framework domain at all. Although the applications applying a particular software framework are from a common domain, the requirements to these concrete applications can be very diverging. Consequently, a process model for the considered framework domain could possibly describe the application development only on a very high level.

Lightweight process models typically only define a few artifacts and require few management effort. When evolving a software framework, its design and implementation is in constant change. All artifacts describing the framework and its functionality must be revised again and again. Consequently, it is not reasonable to write all artifacts documenting the framework in parallel to its development. Thus, to reduce effort to evolve the framework, only as much documentation should be written, as it is required to complete the documentation when finishing a framework development milestone, e. g., an official release of the framework for usage by external organizations (cf. [CC02]). Consequently, using a lightweight process model provides for an efficient management of evolving software frameworks at less cost. In the following section, we present our ProMoCF approach, a lightweight process model for developing component frameworks.

7.2 Modification of Pree's Approach for Object-oriented Frameworks towards Component Frameworks

The motivation for modifying Pree's hot-spot-driven approach for object-oriented frameworks towards a process model and development method for component frameworks is twofold [SB05a, SB06]. These both arguments for modifying the original approach are presented in the following:

- With the original hot-spot-driven approach aCiRC-cards (abstract Class/interface Responsibility Collaboration cards) are used to identify the abstract classes of the framework's object model. Also framework clusters are defined on the basis of key abstractions (see [FPR02]). Both could be useful to identify the framework's components, however after defining them, they are unconsidered during the remainder of the process. To provide support for developing component frameworks, an explicit activity for identifying the framework's components is needed.
- From our experience using the hot-spot-driven approach, it became apparent that the guideline in respect of the granularity of hot-spots of *about one method* (see [Pre95a]) is not precisely enough. It is not clear, how much and which functionality actually corresponds to each hot-spot. Therefore, we define the granularity of hot-spots and hot-spot-cards, respectively, to be *exactly*

one (hook) method in the programming language. We support this decision by introducing so-called group-hot-spot-cards. A group-hot-spot-card comprises an arbitrary number of hot-spot-cards of the granularity of one hook method. With the original Pree process, the number of hook methods that belong to one hot-spot is only implicitly defined by the respective hot-spot-card. However, the concept of group-hot-spot-cards allows for explicitly determining the number of hook methods by the number of their hot-spot-cards. These group-hot-spot-cards are used to identify the framework's components and to specify the flexibility requirements of these components. In addition, with defining the hot-spots' granularity to represent exactly one hook method, the framework developers have to be accurate with defining their hot-spots' functionality.

In the following, the ProMoCF approach (short for "Process Model for Component Frameworks") is presented. It is the result of our modification of the original Pree approach towards a lightweight process model and development method for component frameworks [SB05a]. Figure 7.1 depicts an overview of the ProMoCF approach. The single activities of this process model are described in the following sections.

7.2.1 Definition of a Specific Object Model

With the component-based software development and the process models we find in this area, e. g., [CD01], [Som04], [Nin96], and [DT03], the development of component-based applications typically begins with defining and specifying the components of the application. Thus, applying these process models, first the components and their interfaces are determined. Then, the object-oriented design and implementation of the inside of the components, i. e., the development of component instances (also called component objects [CD01, p. 6]) is conducted. This procedure, although successful for the development of traditional component-based software, is not applicable for developing component frameworks. Thus, in contrast to the process models for component-based software development, our ProMoCF approach for developing component frameworks begins with creating an initial object model of the framework (like it is conducted for object-oriented frameworks with the original Pree process described in Section 4.2.2.1). This initial object model is created on basis of the object models of example applications in the framework's domain.

Starting with an initial object model in order to develop a component-based software architecture is quite unusual. However, this approach is necessary as the components needed for the framework are not known in advance. Thus, the framework components cannot be specified like with the traditional process models for developing component-based applications. There is only the chance to determine the framework components by analyzing the object models of already existing applications in the framework's domain, building an initial object model for the framework on basis of these applications, and then to stepwise abstract from this object model in order to evolve the functionality of the component framework. By this, also the appropriate framework components emerge.

Thus, with the ProMoCF approach, we first conduct an object-oriented analysis of the future inside of the component framework, without knowing the distinct framework components that will emerge. Only by stepwise abstracting and evolving

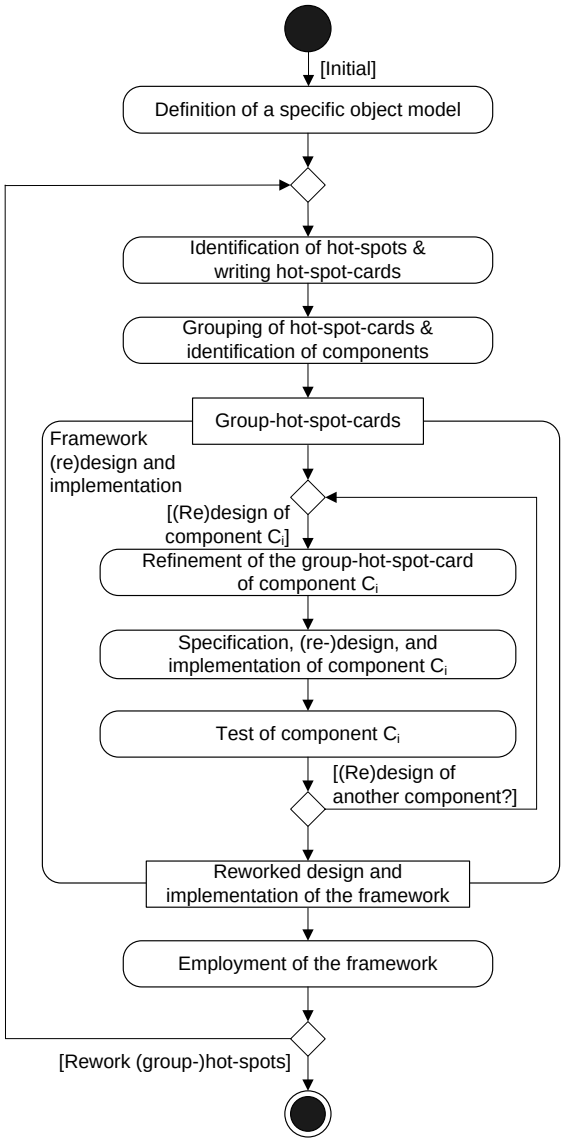


Figure 7.1: The hot-spot-driven lightweight process model for component frameworks

from the initial object model, we can identify and determine the individual framework components and receive the framework's specification.

For creating the initial object model, the models of the existing example applications in the framework domain are accumulated. If the considered example applications already base on component technology, the specific object model is created by accumulating the object models of the example applications' components. In this

case, also an initial component model can be created. The initial component model can help identifying the framework's components.

Using an initial object model to create a component framework entails some disadvantages: Existing components cannot be used as input to the ProMoCF approach without dismantling them into their basic bricks. Thus, the divide-and-conquer-principle of component technology cannot be applied. As a consequence of starting with an initial object model, only such systems can be considered that are limited in regard of their size and complexity and for which a (flat) object-oriented analysis and modeling approach is in principle feasible and can be conducted with a reasonable effort.

On the other hand, there are some advantages when employing an initial object model (actually a class diagram) for developing a component framework: With employing appropriate re-engineering tools, the source code of the existing applications in the framework's domain can be analyzed and reused by generating class diagrams from it. This allows the framework developers to directly fall back on the existing applications for designing the framework. The created class diagrams are employed to identify the framework's hot-spots and to insert these hot-spots into the framework. Creating an initial object model first and then abstracting from it is a reactive approach (see Section 7.1) allowing to exploit the knowledge and expertise of the existing applications in the domain. The framework developers can exploit this knowledge and (stepwise) abstract the framework's functionality and identify the framework's components from the initial object model.

7.2.2 Identification of Hot-spots and Writing of Hot-spot-cards

As the original Pree process, the iterations of the process model's main cycle begin with the identification of hot-spots and writing of hot-spot-cards. As introduced in Section 4.2.2.1, hot-spot-cards constitute a simple but effective means for documenting and communicating the flexibility requirements to the framework between the domain experts and the framework developers. By this, hot-spot-cards bring the specific knowledge of the domain experts into the framework development process. For examples of hot-spot-cards see [Pre96a, Pre99].

7.2.3 Grouping of Hot-spot-cards and Identification of Components

In contrast to the original Pree process, the hot-spot-cards are arranged into logical groups and so-called *group-hot-spot-cards* are written. Arranging the hot-spot-cards into logical groups requires a lot of cognitive work by the framework developers. It is the most subtle activity of the ProMoCF approach. For grouping the hot-spots, the flexibility requirements described on each hot-spot-card are compared. Then, the hot-spot-cards are arranged into logical groups.

For arranging the hot-spot-cards, it might seem reasonable at first glance to arrange the hot-spot-cards into groups dealing with similar services. However, it is not reasonable to put similar services into one group-hot-spot as this will not lead to reasonable framework components. For example, arranging the service for opening a data connection to a user profile server with the service for opening a data

connection to a media data server will probably not yield in a reasonable framework component.

Rather than arranging hot-spot-cards describing similar services, a criteria for arranging hot-spot-cards into logical groups is the hot-spot-cards' cohesion predefined with the framework's domain. This means that those hot-spot-cards are grouped that have a high affinity in regard of solving a particular problem in the framework's domain (cf. eating the elephant pattern in [CC02]). Thus, those hot-spot-cards should be arranged into logical groups that describe flexibility requirements for a *common problem*. Another good indicator for arranging hot-spot-cards into logical groups is to identify those cards that work on the same data. These hot-spot-cards are likely to belong to the same framework component.

For example, within a framework for managing files and folders, one would probably group the hot-spot-card to *Add an arbitrary document to a folder* with its counterpart, the hot-spot-card *Remove an arbitrary document from a folder*.[1] The corresponding group-hot-spot-card could be called *Manage arbitrary items in a folder*. The framework could then call the files in the folders to obtain their size in bytes, independently of whether they are a text file, program file, or of any other domain-specific file type. This could be used to accumulate the file sizes over arbitrary folders and types of files.

For each logical group of hot-spot-cards dealing with the flexibility requirements for a common problem, a corresponding group-hot-spot-card is written. Basing on this analysis and arrangement of hot-spot-cards, it seems reasonable to realize each of the logical groups written down on the group-hot-spot-cards as a self-contained framework component. By this, the modeling aspects of the framework's domain are separated into different framework components (cf. [CC02]).

Consequently, the initial object model of the framework is partitioned into the different framework components defined by the group-hot-spot-cards. Such a partitioning of a software system into bounded units of analysis is necessary in general when the system becomes too large for a global analysis to be feasible [SGM02]. However, it is especially important when a system is meant to be independently extensible [SGM02] such as a component framework. Thus, with the ProMoCF approach, the initial object model is partitioned with the group-hot-spot-cards into reasonable units of analysis. The partitioning activity for finding reasonable framework components is crucial and has a large impact to the success of the developed component framework (cf. [SGM02, p. 139]). Only a good partitioning into reasonable components allows us to construct a complex software system such as a component framework (cf. Szyperski et al.'s units of analysis [SGM02, p. 141]). As hot-spot-cards capture framework flexibility requirements, the group-hot-spot-cards define and specify the flexibility requirements to the framework components. By this, the ProMoCF approach not only provides for identifying the framework's components, but also for developing flexible instances of the framework components. Flexible instances means that the framework components are prepared to be specialized in the traditional object-oriented framework way, i. e., that they are developed by using an object oriented framework. As we will see in Section 8, each

[1] Note: In principle, hot-spot-cards can be divided into data-hot-spot-cards and function-hot-spot-cards as described in Section 4.2.2.1. To be more precisely, the presented hot-spot-cards are data-hot-spot-cards (see [Pre96a]).

group-hot-spot-card identified for our MM4U framework corresponds to one component within this framework and is responsible for conducting a specific task within the general multimedia personalization process.

Grouping of hot-spot-cards to logical groups provides a means for abstracting from some flexibility requirements to the framework if they are not necessary to discuss in detail. For example, not every single hot-spot-card needs to be discussed with the domain experts. However, it is important that the software developers know which hot-spot-cards and corresponding flexibility requirements are hided by the group-hot-spot-cards.

By building logical groups of hot-spots and writing them down on group-hot-spot-cards automatically a coarser and with it a more concise view to the framework's flexibility requirements is created. The hot-spot-cards can be attached to their corresponding group-hot-spot-card, e. g., by using a paper clip. Also new group-hot-spot-cards can be added in this activity for which no corresponding hot-spot-card could be determined in the previous activity so far, as it is not clear which hot-spots this group-hot-spot-card might have. Finally, group-hot-spot-cards can be build of other group-hot-spot-cards to provide an even more abstract view. As a consequence, nesting group-hot-spot-cards means to build a hierarchy of framework components.

The structure of group-hot-spot-cards is similar to the structure of hot-spot-cards. They have a name, a one-sentence description of what the group-hot-spot-card is about, a brief description of the group-hot-spot-card's behavior for at least two concrete situations, and two check boxes for determining whether adaptation during runtime is necessary and whether a configuration tool for end users is needed. Optionally, a list of the attached hot-spot-cards is added. Examples of group-hot-spot-cards written for our component framework MM4U are depicted in Figure 7.2.

7.2.4 Specification, (Re-)Design, and Implementation of the Framework Components

In this activity, the specification, design, and implementation of the component framework is conducted. The major difference to the original hot-spot-driven process is that it has sub-activities for developing framework components. For each component that shall be (re)designed and improved in the current main cycle iteration, the sub-activities as described in the following are conducted.

So far, not all hot-spots are necessarily already identified and additional, previously undiscovered hot-spots can emerge. Therefore, in the first sub-activity of the (re-)design and implementation for the framework components, further hot-spots of the component's group-hot-spot-card are identified. This results in a refined group-hot-spot-card.

In the next sub-activity, the component's interfaces are specified and an instance of the framework component is designed and implemented. For it, the group-hot-spot-cards provide input for specifying both the "outside" and the "inside" of the components. They support the application developers in determining the component's provided interface as well as the component's realization (and with it implicitly the required interface of the framework component). The provided interface [OMG05] describes the services offered by a component [RS02]. The realization [OMG05] determines the framework component's internal behavior, i. e., its implementation

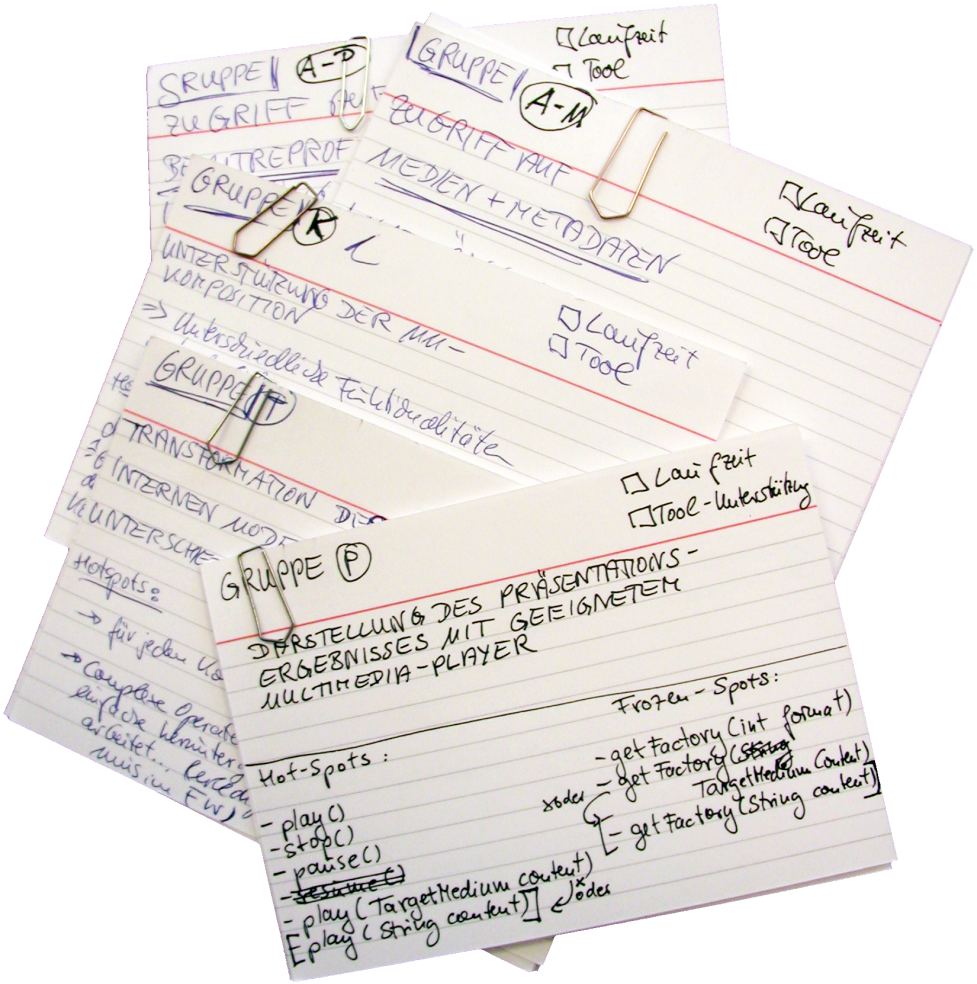


Figure 7.2: The group-hot-spot-cards of our MM4U component framework

(cf. [CD01]). The required interface [OMG05] specifies which other external services and components shall be employed for realizing the component's internal implementation, i. e., what services are needed by a framework component to fulfill its own functionality [RS02]. A detailed description of the provided interfaces, required interfaces, and realization of software components will be presented later in Section 8.5, when it comes to the actual specification, design, and implementation of the components of the MM4U framework.

For the design and implementation of the framework component instances, each hot-spot-card of the group-hot-spot-card is considered and it is determined, whether the flexibility requirement described on it constitutes a service that shall be part of the provided interface of the framework component or if it is for the internal realization only. For example, consider a group-hot-spot-card specifying the flexi-

bility requirements for providing accounting data stored in an arbitrary database. Some hot-spot-cards of this group-hot-spot-card could describe the service to open and close the database connection and read and write some accounting information. These hot-spot-cards should clearly be part of the provided interface of the component. However, another hot-spot-card could specify some abstract functionality to map the internal data structure of the concrete accounting object to the features of the used database. Such a flexible mapping functionality is necessary when different concrete databases can be used for implementing an accounting data component instance. This mapping functionality provides valuable information for the realization of the component. It should be clearly not part of the provided interface.

For specifying the component, the framework developers need to decide which of the hot-spot-cards are associated to the provided interface and which to the realization. This can also be determined prior to the specification, (re-)design and implementation activity by stating the corresponding interface on the hot-spot-card.

As the description and example above shows, the contribution of the group-hot-spot-cards is twofold. First, they provide for describing the provided interface of the framework components. Second, they also provide information for the realization determining how to design and implement a flexible instance of the framework component. Besides the provided interface and the realization, it is also very important to specify the required interface [OMG05] of the framework components. For example, a framework component for analyzing accounting data within a component framework for financial software will probably require access to data stored in some databases. Consequently, for the required interface of this component it is specified that an appropriate component for storing and retrieving accounting data needs to be available. These dependencies between framework components are provided by the framework's domain.

The design and implementation of the framework component instances can be revised and improved in the following main cycle iterations. Implementing the framework components' instances also includes writing the documentation that is needed to use the components. Then, the concrete implementations of the framework component instances are tested in the next sub-activity in an appropriate test environment, e. g., by applying the automatic test approach.

Since not all components are improved in each iteration, they can have different maturity (cf. clusters in [FPR00]). For example, while some components are already in the implementation phase, others can still be in the requirements or design phase. In addition, the development of the framework components can also be concurrently conducted (not shown in Figure 7.1). The result of this activity is a (revised) design and an (improved) implementation of the framework components.

With the ProMoCF approach, well-designed framework components emerge. The framework component instances are designed with the group-hot-spot-cards as traditional object-oriented framework. Consequently, the framework component instances are prepared to be adapted to the requirements of the different concrete applications in the traditional object-oriented framework way. The framework components specification shield the single components instances and their traditional object-oriented frameworks and encapsulate them into independently deployable and employable units.

7.2.5 Employment of the Framework

In this activity, the (revised) framework is tested and evaluated in regard of the quality and reusability of its design. The only way to identify errors and weak points in the framework's design is to develop concrete applications that use the component framework [BMM+99]. Weak points in the design of the component framework are those where the selected hot-spots are inappropriate (cf. Section 4.2.2.1). This means that parts of the framework's design are too static and thus additional hot-spots or even group-hot-spots need to be added. Or some parts of the design are too flexible such that the component framework does not provide for the desired standardized architecture for the considered domain. To provide for a more fixed architecture, some framework hot-spots need to be modified or removed. Consequently, the component framework is iteratively improved and the framework as whole is evolving.

Developing a concrete application by applying the evolving component framework means composing instances of the framework components (that meet the requirements of the application's domain) according to the rules defined by the framework. For specifying and identifying these components and for developing the concrete application, one of the standard component-based software development processes such as [Som04] or [Bos00] can be applied. The framework component instances can be adapted to the requirements of the concrete application by, e. g., modifying existing ones, reusing them in different configurations only, or developing new ones. The more concrete applications are developed by using the component framework, the more different concrete instances of the framework components exist that can be used for composition and reuse in other applications. As the ProMoCF approach provides for developing flexible framework component instances by designing and implementing them as traditional object-oriented frameworks, they can be reused and adapted to the requirements of the concrete applications by means of traditional object-oriented technology. Errors in the component framework and the concrete applications employing the framework are corrected immediately. Weak points in the framework design are improved in the next main cycle iteration. In regard of documentation, only as much as needed to develop the framework and the concrete applications is written. Only for finishing a framework milestone, i. e., a release of the framework to an external organization, a complete documentation is written.

Like with the regular component-based development where the developer team should be divided into two parts, one part providing the components and the other part using the components, the development team of component frameworks shall be divided. Here, one part of the developer team provides the component framework and the other part uses and applies the framework for the development of concrete applications. In general, this approach is useful to apply when reuse of the developed software artifacts is important.

7.2.6 Criteria for Repeating the Process Model's Main Cycle

As described in the previous section, some of the identified group-hot-spots and their associated hot-spots may emerge to be inappropriate when developing concrete applications employing the just developed component framework. Then, the main cy-

cle of the ProMoCF approach is repeated and the weak parts of the framework's design, i. e., the corresponding group-hot-spot-cards with their attached hot-spot-cards are reworked. These iterations of the main cycle are repeated at least as long as there are still some design improvements necessary for the framework. Typically, many iterations of the main cycle are required until a well-designed component framework emerged that can be released.

However, also after releasing an initial version of the framework, the development of the framework with the ProMoCF approach does not stop. For example, further iterations of the main cycle are conducted to improve and update the framework. Such an improvement will lead to a second, third, and so on release of the framework.

In addition, further iterations of the main cycle are required, when the domain of the component framework changes over time. With a changing domain also the requirements to the corresponding framework change and need some rework. To keep the framework up to date, i. e., to track possible changes of the domain and the requirements to the framework, additional main cycle iterations of the ProMoCF approach are conducted. Within these iterations, the group-hot-spot-cards and their hot-spot-cards are modified to the new requirements and the framework is adapted accordingly.

7.3 Applicability and Limits of the ProMoCF Approach

The ProMoCF approach is not an all embracing process model for the development of component frameworks. It is not the *silver bullet* solving all problems and challenges of developing and evolving component frameworks. In the following, it is described for what kind of software family the ProMoCF approach is applicable. As Prechelt motivates with his claim for a Forum for Negative Results [Pre97a], it is not only important when a software engineering approach is applicable; it is also as much important to know when it does not work. Consequently, the limits of the ProMoCF approach will be drawn. Both will help potential applicants of our approach to decide whether ProMoCF is appropriate for their project.

The ProMoCF approach yields at developing component frameworks that are targeted at a specific domain. Input to this component framework development process is an initial object model of the framework. This object model is accumulated by the object models of existing applications in the domain of the framework. For it, existing applications in the domain are analyzed. If an object model is small enough to be conveniently handled and analyzed with merely object-oriented technology, this technology might be the best choice for implementing the system. However, if an object model is becoming too large for a global analysis and an object-oriented implementation is becoming to be practically infeasible, component-based technology needs to be applied. This is especially important if other requirements are made such as partitioning the object model into major aspects of the considered domain with the goal to obtain reusable and independently deployable and extensible units. One of the central non-functional requirements to the MM4U framework is to be independently extensible in regard of integrating existing solution approaches and systems at different levels for providing application-specific multi-

media personalization functionality. In addition, the functionality provided by the single MM4U framework's components should also be independently deployable. To provide support for such problems, the ProMoCF approach is designed for. It is applicable for software systems, for which in principle an initial object model can be defined. However, this object model is too large for a global analysis and its implementation as object-oriented framework would be impractical as it is very difficult if not unfeasible to independently deploy and extend the framework (see the fragile base class problem that emerges with composing object-oriented frameworks in Section 4.1.2.1). With the ProMoCF approach's group-hot-spot-cards, the initial object model is partitioned into reasonable units of analysis and implementation. Each group-hot-spot-card is aimed at specifying and is resulting in a distinct framework component solving a specific task in the considered framework domain. By the nature of component technology, the framework's components as well as the component framework itself are independently deployable and extensible units. Consequently, the ProMoCF approach serves for considering Szyperski et al.'s different aspects of scale and granularity for designing software components in regard of the different units of analysis [SGM02, 140ff.].

In the case that the considered existing applications already base on component technology, the components of these applications have to be dismantled into their object models first. Only by this, existing component-based applications can be taken into account for the development of the component framework. As a consequence, the ProMoCF approach does not provide for COTS-reuse (short for "commercial off-the-shelf reuse") of existing components in the domain.

As the ProMoCF approach is a lightweight process model, its application is heavily limited to software projects with a small development team only. Otherwise it will not be possible to keep all members of the development team up to date with the design decisions and development progress of the framework.

7.4 Summary

With the ProMoCF approach we presented a modification of the hot-spot-driven approach for object-oriented frameworks by Pree towards a process model for component frameworks. The ProMoCF approach provides methodical support for identifying framework components and specifying the flexibility requirements to these components. This is achieved by introducing the concept of group-hot-spot-cards.

For evaluating the ProMoCF approach, it has been applied for developing and evolving the MM4U component framework. Thus, the MM4U framework can be considered as a case study of the proposed development process. While the ProMoCF approach has been evaluated by applying it for the development of the MM4U component framework, the component framework itself has provided important feedback to the development process. By this a mutual benefit between improving and maturing the ProMoCF approach and developing and evolving the MM4U component framework emerged.

To ensure the quality and to prove that the developed framework is well-defined, the ProMoCF approach defines in its activity *Evolution of the framework* to employ the created component framework for the development of concrete applications. Thus, the ProMoCF approach is not only applied for the development of the MM4U

component framework. Also the outcome of the ProMoCF approach, i. e., the MM4U component framework itself is tested and applied for various demonstrator applications in different domains. The developed demonstrator applications provided for improving the design of the MM4U framework and ensure that the outcome of applying the ProMoCF approach is an applicable and well-designed component framework. In addition, with developing the demonstrator applications we obtained feedback for maturing the ProMoCF approach.

In the following Section 8, the design and implementation of the MM4U component framework along the single activities of the lightweight process model ProMoCF are presented. In Section 9, the impact of personalization to the software development process in general and the design of multimedia applications in particular are considered. In addition, the MM4U framework support for developing personalized multimedia applications is described. The employment of the MM4U component framework in multiple applications in the domain of multimedia personalization is presented in Section 10.

8 Development of the MM4U Component Framework

In this section, we present in detail the experiences gained with applying the ProMoCF approach for developing and evolving the MM4U framework and its components. This includes the basic organization of the framework, represented by its components, and the relation of the components to each other and to the environment. By applying the ProMoCF approach for developing and evolving the MM4U component framework, we conducted an evaluation of our process model and development method for component frameworks. The goal here is to develop a well-defined and applicable component framework. However, it is only the first step in evaluating the ProMoCF approach. The second step in evaluating the ProMoCF approach is the development of concrete applications on the basis of the MM4U framework. By this, the outcome of the ProMoCF approach, i. e., the quality of the MM4U component framework itself is evaluated. The concrete applications using the MM4U framework are presented in Section 10.

According to the design and implementation requirements in Section 5.2.3, the development of the MM4U component framework is conducted in the programming language Java and by applying a sophisticated IDE. In the following sections, the development of the MM4U component framework is described along the different tasks and activities defined by our process model and development method ProMoCF presented in Section 7. We start with the definition of an initial object model for the MM4U framework in Section 8.1. This object model is determined on the basis of prototypical applications as well as the related work in the field of multimedia personalization. In Section 8.2, the identification and writing of hot-spot-cards and group-hot-spot-cards is described, holding the flexibility requirements to the MM4U framework. The group-hot-spot-cards lead to the identification of the framework components as presented in Section 8.3 and the definition of the MM4U framework component architecture introduced in Section 8.4. The general issues involved with specifying, (re-)designing, and implementing the MM4U framework and its components are presented in Section 8.5. In Section 8.6, the underlying component-based software engineering concepts for developing the MM4U framework components and their instances are introduced, before the development these components and instances are presented in detail in Sections 8.7 to 8.12. Other design issues of the MM4U framework in regard of performance and a common exception concept for the framework are briefly described in Section 8.13. In Section 8.14, the general usage of the MM4U component framework by a concrete personalized application is presented. In Section 8.15, the deployment of the MM4U framework components are described. With mobileMM4U, a specific subset of the MM4U framework targeted

at mobile devices is presented in Section 8.16, before this section concludes with a short summary.

8.1 Definition of a Specific Object Model

For defining the initial object model of our MM4U framework, we analyzed the source code of three prototypes developed in the area of personalized multimedia applications. These first prototypes are a personalized sightseeing tour through Vienna [Bol03b] and Oldenburg [BK03], a GPS-based mobile paper chase game [BKW03, KKRB04], and a personalized music news letter, generating personalized multimedia music news letters in HTML and SMIL in regard of the user's favorite genres and artists [Ric03].

Having the possibility to analyze three object models of related applications is a quite unusual situation, as the development of such object models itself is a complex task. The object models of these prototypes were studied in full depth in regard of similarities and dissimilarities. On basis of the analysis of our prototypes as well as the extensive study of related systems and solution approaches for creating personalized multimedia content (see Section 3.2 and 3.3), we defined an initial object model for the MM4U framework.

8.2 Identification and Writing of Hot-spot-cards and Group-hot-spot-cards

According to the ProMoCF approach, an initial set of hot-spots has been identified and corresponding hot-spot-cards have been written. For identifying the hot-spots, our experiences gained in developing the prototypes of personalized multimedia applications as well as the analysis of related work in the area of generating personalized multimedia content have been helpful. The initial hot-spot-cards were arranged and re-worked by conducting several iterations of ProMoCF's main cycle. This identification of hot-spots and the (re-)arrangement of the corresponding hot-spot-cards led to five logical groups. Correspondingly, five group-hot-spot-cards were written. Each of these five group-hot-spot-cards addresses a particular task of the general multimedia personalization process as described in Section 5.1. Thus, the identified group-hot-spots with the associated hot-spots constitute the domain model of the framework. The identified group-hot-spot-cards and their task within the multimedia personalization process are described in the following.

- The first group-hot-spot-card specifies the flexibility requirements needed for integrating existing systems and solutions for storage, retrieval, and access to user profile information and context information.
- The second group-hot-spot-card specifies the flexibility requirements needed for integrating existing systems and solutions for storage, retrieval, and access to media elements and their associated meta data.
- The third group-hot-spot-card specifies the flexibility requirements for composing arbitrary personalized multimedia content in the internal multimedia content representation model.

- The flexibility requirements for transforming the multimedia representation model to the syntax and features of different (standardized) multimedia presentation formats are written down on the fourth group-hot-spot-card.
- Finally, the fifth group-hot-spot-card specifies the flexibility requirements for integrating different existing player software for multimedia presentation.

During the iterative framework development process further hot-spot-cards were identified. These hot-spot-cards were attached to one of the existing group-hot-spot-cards identified for the MM4U framework above. In addition, some of the identified hot-spot-cards were redefined and others were removed. For all group-hot-spot-cards and their subsidiary hot-spot-cards neither an adaptation of the functionality during runtime nor a software tool to support end users with adapting the functionality is necessary.

The five group-hot-spot-cards emerged during the identification and rework process of the hot-spot-cards. In the beginning of the development process of the MM4U framework it was not clear how many group-hot-spot-cards would eventually be necessary. However, with continuously developing and evolving the framework and applying it for the development of concrete applications, the number of group-hot-spot-cards finally meet to five.

8.3 Identification of the Framework's Components

The group-hot-spot-cards divide the flexibility requirements to the MM4U framework in five logical groups, each addressing a particular task in the multimedia personalization process. Following our ProMoCF approach, we defined for each group-hot-spot-card a distinct component within the MM4U component framework. On basis of the group-hot-spot-cards, we specified the interfaces of the framework components and implemented a concrete instance of each component.

At the beginning of the development of the MM4U framework there had been some imagination of the flexibility requirements and demands to be made to the component framework. However, it was not clear, how many components would be necessary to realize these requirements and how to reasonably divide the flexibility requirements into components. By identifying the hot-spot-cards and arranging them to the five group-hot-spot-cards, we are sure that all necessary framework components have been identified. Another framework component cannot emerge, as all requirements to the framework (defined with the hot-spot-cards) have been successfully mapped to the identified framework components. Even more important is, however, that we are sure that the emergent components are the right ones, i. e., that the flexibility requirements to the framework are reasonably divided into (the identified) components.

8.4 Definition of the Framework's Component Architecture

In Section 5.3, we presented a functionality-driven layered architecture of the MM4U framework. The different layers of the framework provide modular sup-

port for the different tasks of the personalization process of multimedia content as shown in Figure 5.1. The following Figure 8.1 also shows this architecture. However, the five components of the framework are indicated by five boxes surrounding the framework's layers. The MM4U framework layer (3) to (5) each constitute a single software component within this architecture. However, the Accessor layer and Connectors layer do not. Instead the left side and the right side of the layers (1) and (2), i. e., the User Profile Accessor and User Profile Connectors as well as the Media Pool Accessor and Media Data Connectors, each form a distinct software component in the MM4U framework.

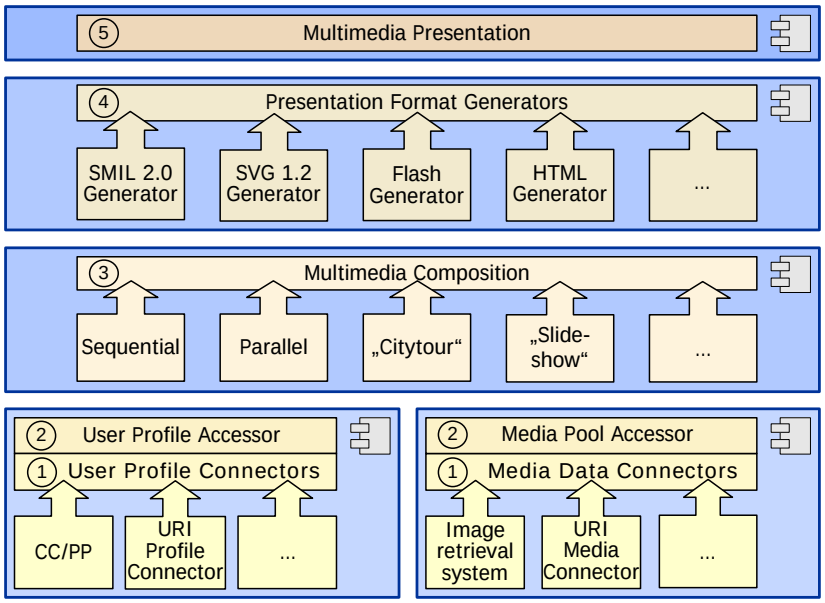


Figure 8.1: Combined functional and component view of the MM4U framework

Consequently, Figure 8.2 depicts a UML component diagram of the MM4U framework. The diagram shows the five components of the MM4U framework and the components' central interfaces of the provided and required data, respectively. The two components at the bottom of the figure depict the access and connectors to user profile information and media data. The central interfaces here are IUserProfile for the user profile information and IMedium as well as IMediaList providing a medium element and set of media elements together with their associated meta data. The user profile information and media data is exploited for the multimedia composition task in the Multimedia Composition component. This component provides an object-oriented representation of the generated personalized multimedia content tree. The root node of this multimedia content tree, which is of type IVariable, is passed to the Presentation Format Generators component. Here, the content is transformed into the concrete multimedia presentation formats. The multimedia presentation in final presentation format is passed as object of the interface IMultimediaPresentation to the Multimedia Presentation component. This component actually renders and displays the presentation on the end device of the user.

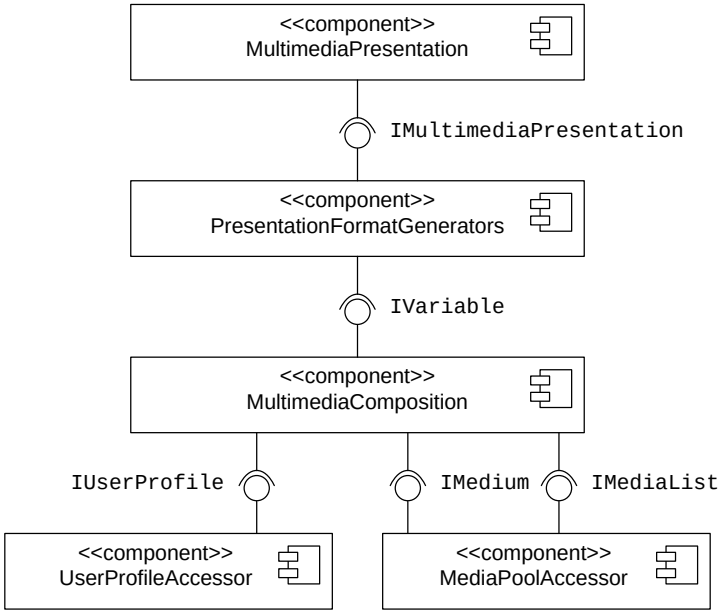


Figure 8.2: UML component diagram of the MM4U framework architecture

8.5 Specification, (Re-)Design, and Implementation of the MM4U Framework and its Components

The framework, its components, classes, and interfaces, are specified using the UML. The implementation of the framework component instances is carried out in Java (using the JDK 1.4 and JDK 1.1.8, respectively, for the mobile variant of the framework). As we applied the ProMoCF approach for developing the MM4U framework, we implemented a first prototype of the framework which then has undergone constant review, re-design, and re-implementation iterations for a stepwise refinement and enhancement of the framework's components. This redesign was triggered by the experience gained with employing the framework for the development of several demonstrator applications to prove the applicability of the MM4U framework in different application domains (see Section 10 for demonstrator applications). The demonstrator applications where used as valuation of the framework. They provided feedback in regard of the comprehensiveness and applicability of the framework to improve it in subsequent iterations. The demonstrator applications pinpointed the weak points of the MM4U framework's architecture, i. e., they identified those parts of the framework design where it was too inflexible and additional hot-spots needed to be added. In addition, also those parts of the framework design with too much flexibility were identified such that the control flow of the application was unclear and ambiguous to the application developers and additional frozen-spots needed to be added.

Many iterations have been conducted for (re)designing and improving the MM4U component framework. During these iterations, the single components had

different maturity. For example, during redesign of the Media Pool Accessor and Connectors component, the implementation of the Multimedia Composition component has been finished. And while implementing the Multimedia Composition component, the Presentation Format Generators component was still in its initial design.

For the (re)design and implementation task, we employed the Eclipse [Ecl06] IDE. This IDE provides powerful refactoring functionality, which is very useful for the design and implementation of a software framework. Without applying such a sophisticated IDE, the development of the MM4U component framework would probably not have been feasible due to the many redesign iterations and refactoring sessions that were necessary.

8.6 Development of the MM4U Framework Components and Component Instances

Before we present the specification of the framework components and the development of the component instances in detail, we introduce the underlying component-based software engineering concepts that are used as basis for specifying the components and developing the instances. In addition, we introduce how these component-based concepts are mapped to the development of the MM4U framework in the object-oriented programming language Java.

When developing software components, two parts of the components are to be considered [WRBS05, section 3.3], the "outside" and the "inside" of a component. The "outside" of a software component describes the component's behavior for usage by other components as well as the services and components required to fulfill its own services. It is characterized by the preconditions and postconditions of the component's contract. Depending that the preconditions of a component are fulfilled by offering the right execution environment, the component offers its services as described in the postcondition [RS02]. The services offered by a component are specified in the provided interface [OMG05, RS02]. Thus, the provided interface determines the postconditions of a component [RS02]. The provided interfaces of the components of the MM4U framework, i. e., the services the framework components offer are basically specified and described by employing Java interfaces. As Java interfaces do not allow for specifying the order in which the services of a component can be used, i. e., they do not allow for specifying the provided protocol of a component, finite state automata (FSA) are applied here. The provided protocol of each component of the MM4U framework is determined by using a FSA. The FSA specifies the allowed call sequences to the component's services. The notation for the FSA is taken from [Reu01b].

Which services from other components are used by a component to actually realize and provide its own services are described in the requirements part of the component's specification, i. e., they are specified in the required interface [OMG05]. Thus, the required interface defines the preconditions of a component [RS02]. For example, an application-specific component needs to store some information in a persistence system, e. g., a database. This storage functionality can be specified in the required interface. Any persistence component implementing this interface can

then be used by the application-specific component to actually realize its storage functionality.

As the preconditions of a component might not be fully satisfied by a specific execution environment and the postconditions of a component are not necessary to their full extend by all components using it, also subsets of the preconditions and postconditions can be modeled in the component's contract describing the component's behavior to its "outside" world. Such a component contract that can be adapted to different specific execution environments of a component is called parameterized contract [RS02, Reu01b, Reu01a].

The "inside" behavior of a software component is described by the realization [OMG05]. It can be considered as the private design of the component, describing how the component fulfills its provided interface [ABB+02]. To model the components internal behavior, different approaches can be applied, e. g., formal methods such as state chart diagrams, petri-nets, pi-calculi, and algebras. As the formal methods tend to be complex and difficult to handle, we use semi-formal and informal methods for modeling and describing the internal realization of the MM4U framework's components. Here, textual descriptions, sequence diagrams, and UML class diagrams are employed. Textual descriptions introduce the component's behavior on a mere verbal level. However, they aim to be easy to understand and compact in their length. Sequence diagrams are used to depict communication flows. UML class diagrams are employed to specify the internal design of a component instance on a mere functional level. However, they provide for defining the general structure of a component and are a good means for developing component instances. The realization describes the internal behavior of a component and can require some services that are provided by other components. Thus, the provided interfaces of the other components are employed and used to realize the component's functionality. As a consequence, the required interface can be associated to both the inside and outside of a component. As the required interface serves as basis for realizing the component's provided interface, i. e., for realizing the provided services, it refers to the internal implementation of a component. However, as the required services have to be specified such that they are visible by other external components providing them, they can also be associated to the outside of the component.

The components' provided interfaces and required interfaces are considered as first-class entities. This means that they are independent of the framework's components. The interfaces do not belong to one distinct component only, but can appear at multiple places in the component framework. This means, that the interfaces can be associated to an arbitrary set of the framework components. As a consequence of being first-class entities, the interfaces are stored in a separate place from the framework component instances. This follows the separated interface pattern [FRF+02] postulating the definition of the interfaces in a separate package from their implementation or following Szyperski et al. [SGM02] "separation of the immutable plan from the mutable instances is essential".

The implementation of a software component is often realized by means of an object-oriented framework [Bos00]. As we will see in the subsequent sections, almost all component instances of the MM4U framework components are designed and implemented as traditional object-oriented frameworks. Only the Multimedia Composition component, with its initial design of an object-oriented framework, ma-

tured over time to a flexible and extensible toolbox for composing and assembling arbitrary personalized multimedia content.

The development of concrete instances of the MM4U framework components is a complex and challenging task in itself. Consequently, the development of the components and their instances is described in detail in the following Sections 8.7 to 8.11. The description of each component is structured as follows: First, the functionality and objectives of the component as introduced in Section 5.3 are briefly repeated. Second, the underlying concepts used for developing and designing the component are presented. Third, the contractual specification of the component is presented. As the description of the required interfaces of the MM4U framework components is only of minor importance for the presented development of the MM4U framework, we concentrate here on the provided interfaces. As the provided interfaces are specified in Java, we enrich each with a corresponding FSA describing the sequence of how to use the corresponding component's services. Finally, the design and implementation of a concrete instance of the framework component is described.

A component framework itself should be designed and implemented as single software component. Thus, a sixth group-hot-spot-card is introduced in Section 8.12 for developing the MM4U component. The MM4U component embraces the framework components presented in Sections 8.7 to 8.11. In addition, also the contractual specification of the MM4U component and the development of an instance of this component is presented in Section 8.12.

8.7 Development of the Media Pool Accessor and Connectors Component

For creating personalized multimedia content those media elements have to be selected from the media databases that are most appropriate to the user's profile information. This personalized media selection is realized by the Media Pool Accessor and Connectors component of the framework. It provides concrete applications of the framework access to media elements and associated meta data.

8.7.1 Underlying Concepts

To retrieve those media elements that are of most relevance to the user and the requested presentation, the media pool accessor provides passing the user profile information to the media data connector. An example for user profile information exploited by the Media Pool Accessor and Connectors component is, e. g., for the domain of mobile tourist guide applications information about the user's interests and preferences in regard of the provided sights, the display size of the end device, and the current location of the user. This and further user profile information is exploited by our demonstrator application Sightseeing4U presented in Section 10.1.

To specify additional parameters for querying the media pool storages, further query options can be defined within a query object implementing the Media Pool Accessor and Connectors component's IQueryObject interface. Such a query object is typically created within the Multimedia Composition component (this component is presented later in Section 8.9). The query object is carrying a list of application-specific parameters used for determining the media elements for the multimedia

composition task. For example, the authoring wizard for creating personalized photo albums implemented in our *x*SMART tool (see Section 10.6) can specify a query object such that only those image elements are selected for the photo album that fulfill specific criteria. These selection criteria are, e. g., that the photos must be taken within a specific area and time, are an indoor or outdoor shot, have a minimum and maximum exposure, and possess a sufficient sharpness.

With this query object and the user profile information, the Media Pool Accessor and Connectors component is called by the Multimedia Composition component for retrieving the most relevant media elements. As the Media Pool Accessor and Connectors component aims at integrating and embracing existing multimedia databases and content-based multimedia retrieval solutions, the retrieval of the "best match" can only be left to the underlying storage and management systems. Consequently, within the media data accessor the query object is mapped to the meta data associated with the media elements in the concrete underlying media store. Then, the query object is handed over to the concrete media data connector of the connected media storage solution. The concrete media storage solution determines the result set, i. e., an ordered list of media elements that best match the given query object. Finally, the result set of the query is handed back to the calling component.

8.7.2 Specification of the Component

For the access to media elements and their associated meta data, the Media Pool Accessor and Connectors component defines the narrow provided interface IMediaElementsAccessor shown in Listing 8.1. The provided interface offers services such as openConnection() and closeConnection() for opening and closing a connection to the media storage. The Media Pool Accessor and Connectors component supports active (client-pull) and passive (server-push) retrieval of media elements. For actively querying media elements by a given user profile and application-specific query object as described above, the service getMediaElements(...) is provided. However, also a single medium element can be retrieved by calling the service getMediumElement(...) with the medium element's unique identifier as parameter. To provide support for an automatic notification for new media elements, the Media Pool Accessor and Connectors component offers to add and remove media observers to the media elements connector. For it, the services addObserver(...) and removeObserver(...) are provided.

The provided protocol of the Media Pool Accessor and Connectors component is defined by the FSA depicted in Figure 8.3. First, media element observers can be added (and removed), before a connection to a concrete media elements storage solution is opened and a connection to the underlying media server is established. After establishing a connection to the media storage solution, arbitrary media elements can be retrieved actively by using the corresponding getMediumElement(...) and getMediaElements(...) services. In addition, now the media elements observers start working and notify the personalized multimedia application when relevant media elements occur in the media storage solution. While the media elements connection is established, also further media elements observers can by dynamically added and removed. Finally, the connection to the concrete media storage solution is closed.

```
public interface IMediaElementsAccessor {
  public void openConnection()
          throws MM4UCannotOpenMediaElementsConnectionException;
  public void closeConnection()
          throws MM4UCannotCloseMediaElementsConnectionException;

  public abstract IMedium getMediumElement(String mediumID)
          throws MM4UMediumElementNotFoundException;
  public IMediaList getMediaElements(
          IQueryObject queryObject,
    IUserProfile userProfile) throws MM4UMediumElementNotFoundException;

  public boolean addMediaElementsObserver(IEventListener observer);
  public boolean removeMediaElementsObserver(IEventListener observer);
}
```

Listing 8.1: Provided interface of the Media Pool Accessor and Connectors component

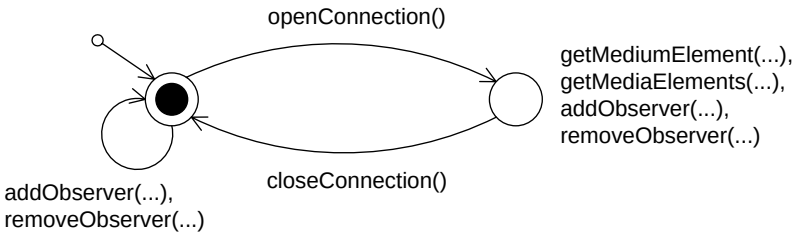


Figure 8.3: Finite state automaton defining the provided protocol of the Media Pool Accessor and Connectors component

8.7.3 Design and Implementation of a Component Instance

The UML class diagram shown in Figure 8.4 depicts the principal design of our instance of the Media Pool Accessor and Connectors component. The abstract class MediaElementsAccessorToolkit implements the provided interface IMediaElementsAccessor. As the suffix Toolkit indicates, the class is implemented by applying the design pattern abstract factory [GHJV04, Mar98]. To create a connector for a particular media elements storage solution, a personalized multimedia application calls the getFactory(*<IMediaElementsConnectorLocator>*) method of the abstract factory with a so-called *locator object* as parameter. Here, the corresponding factory class is instantiated, e. g., URIMediaElementsConnectorFactory, which creates all internal objects needed for an instance of the Media Pool Accessor and Connectors component.

As the parameter IMediaElementsConnectorLocator of the getFactory(...) method shows, we use a slightly modified version of the abstract factory pattern. Instead of passing an integer value for identifying the requested concrete user profile connector, as it is done with the original abstract factory pattern, we use the more flexible mechanism of locator objects. A locator object is a list of properties targeted at holding the parameters necessary to instantiate a concrete media elements connector.

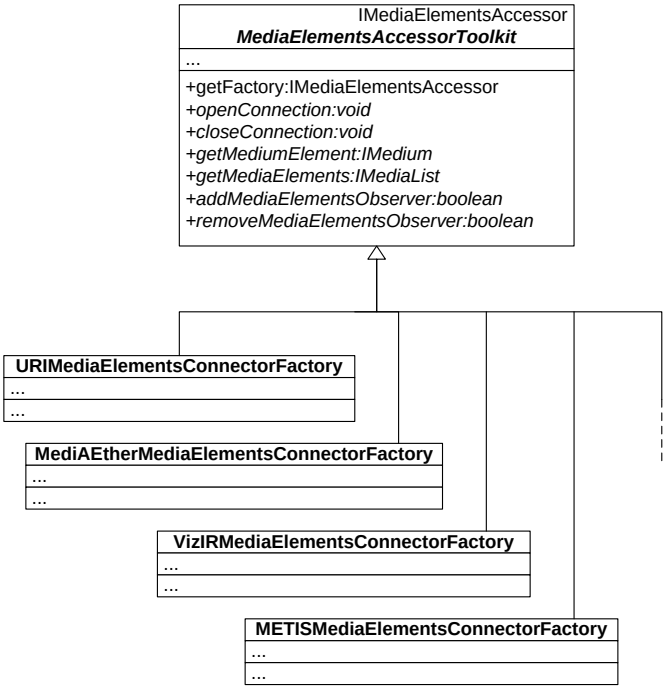


Figure 8.4: UML class diagram of the Media Pool Accessor and Connectors component instance

Therefore, the aim of a locator object is twofold: First, the concrete locator object is used within the getFactory(...) method of the MediaElementsAccessorToolkit class to determine the concrete factory for the corresponding connector. Second, the parameters defined in the concrete locator object are used to initialize the determined media elements connector factory, i. e., the locator object's parameter values are used to instantiate the concrete connector for the specific media elements storage solution.

The more flexible concept of locator objects is necessary, as the parameters required to instantiate the connectors to the different media elements storage solutions can be very different. For example, the parameters required for our concrete media elements connector URIMediaElementsConnector presented in the following Section 8.7.3.1 are the URL of the media server's root path and the name of an index file storing meta data about the media elements. However, a JDBC-based connector to a relational database such as Oracle 10g with interMedia extension [Ora06a, Ora04] for managing media elements such as images, audio, and video elements, would require some different parameters. Here, information about the used driver class and driver name, the database's URL, and the login and password must be provided.

For realizing the automatic notification of personalized multimedia applications when new media elements are available, i. e., to provide for retrieving media elements via server-push mode, we employ the design pattern observer [GHJV04]. This is reflected in the provided interface by the services addObserver(...) and re-

moveObserver(...). Here, media element observers can be added that provide for (possibly application-specific) notification of emerging media elements matching the user profile information.

As mentioned in Section 8.6, the instance of the Media Pool Accessor and Connectors component is designed and implemented as object-oriented framework. The concrete connectors for the specific media storage solutions can be added to the component instance by creating a concrete Factory class of the abstract MediaElementsAccessorToolkit class. In addition, a corresponding locator class must be provided. The control flow within the component instance is held by the MediaElementsAccessorToolkit class and not by its concrete extensions for the specific media storage solutions, provided by the corresponding Factory classes. We developed among others the four concrete media elements connectors shown in Figure 8.4 and integrated them in our Media Pool Accessor and Connectors component's instance. These concrete connectors are described in the following Sections 8.7.3.1 to 8.7.3.4. In addition, another media elements connector has been implemented that integrates an arbitrary set of concrete media elements connectors and thus provides a transparent access to multiple connectors. This connector is described in Section 8.7.3.5.

8.7.3.1 URIMediaElementsConnector

The URIMediaElementsConnector provides a flexible client-pull access to media elements and their associated meta data from the Internet via the http-protocol and ftp-protocol as well as the access to such data from the local hard drive. The meta data of the media elements are stored in a so-called index file. An index file is a text file containing the meta data of the single media elements. For each medium element, it describes the technical characteristics such as width, length, and playback duration and contains the location where to find the medium element in the Internet by a URI. The index file can also comprise any additional information about the media data, e. g., a short description of what is shown in a picture or keywords one can search for. The technical meta data of the media elements can also be automatically extracted from the medium element's media data. Thus, the index file does not need to provide the media element's width, height, and duration.

The URIMediaElementsConnector also provides a set of simple features for image retrieval. For it, the connector supports the exploitation of meta data from an underlying meta data extraction and enhancement architecture for image elements. This meta data architecture supports the extraction of EXIF data [Tec02], which includes among others contextual information about the time, location, and the camera settings when a photo was taken. The meta data architecture also supports for content-based meta data extraction such as calculating the exposure and sharpness of an image element as well as its similarity to other images on the basis of histograms. In addition, it also allows to determine whether a photo is taken indoor or outdoor basing on information about the light condition, daytime, exposure, and if the camera's flash is fired or not. The connector allows for retrieval of image elements according to these meta data. Following a list of features the URIMediaElementsConnector provides.

- Automatic transcaling of image elements, i. e., changing the size of the requested image elements.

- Automatic transcoding of image elements, i. e., conversion between different media formats [LNK04, GT95] such as PNG, JPG, and GIF.
- Support for identifying pictures that are underexposed or overexposed.
- Identification of pictures that are blurred or fuzzy.
- Detection of similar pictures on the basis of color histograms.
- Determination whether a picture is an indoor shot or outdoor shot.
- Provision of clustering information such as temporal relationships to other images.

Media elements are typically referred by today's presentation formats such as SMIL and SVG by providing a URI to the media resource. For accessing transcaled and transcoded image elements, which are not directly accessible by a URI, the URIMediaElementsConnector provides a download servlet. This download servlet enables the access to arbitrary transcaled or transcoded media elements. The download servlet also provides a caching mechanism to increase retrieval speed. For example, once a particular image element is transcaled and transcoded to a specific parameter setting, this manipulated image element is stored in the media cache. The next time, the same media element is requested with the same parameter setting, the original image element is not manipulated again but the manipulated variant stored in the media cache is retrieved instead. Transcaling and transcoding of media elements is necessary when, e. g., a photo taken by a digital camera shall be embedded in a Flash or SMIL presentation targeted at a mobile device. Transcoding of media elements is necessary, when specific media format requirements need to be fulfilled, e. g., to adapt to player specific constraints. For example, the Ambulant Player [BJK+04, CWI05] currently does not support all different versions of PNG-images. Here, the URIMediaElementsConnector can provide, e. g., JPG-images instead.

The features of the URIMediaElementsConnector are extensively used by our demonstrator application Pictures4U, a personalized photo album generator in the web presented in Section 10.5. They are also used by the interactive authoring wizard for personalized photo albums integrated in our context-driven smart authoring tool *x*SMART presented in Section 10.6.

8.7.3.2 MediÆther Event Space

The MediAEtherMediaElementsConnector is an implementation of a media elements connector for integrating the multimedia event space MediÆther [BW03] into the MM4U framework. The MediÆther is a decentralized peer-to-peer infrastructure that allows to publish, find, and notify about any kind of multimedia events. Consequently, the MediAEtherMediaElementsConnector is used to actively notify personalized multimedia applications via the server-push mechanism when new media elements appear in the MediÆther event space. This functionality is used, e. g., by our demonstrator application Sports4U, a personalized sports news ticker presented in Section 10.2. The MediAEtherMediaElementsConnector does not only allow for a server-push brokerage of media elements, but also supports the client-pull retrieval of media elements.

8.7.3.3 VizIR—Visual Information Retrieval Framework

Within the CoCoMA (short for "Content and Context Aware Multimedia Content Retrieval, Delivery and Presentation") project [CTB+05, DE05] of the DELOS Network of Excellence [DEL06] on digital libraries, a VizIRMediaElementsConnector has been developed for integrating the visual information retrieval framework VizIR [EB03, Eid03] into the MM4U framework. The VizIR framework, developed at the Technical University of Vienna (TUV) in Austria, provides a generic architecture for developing visual information retrieval systems [Tec06]. It can be applied for any querying model that bases on the extraction of visual information from visual media elements and the computation of similarities of these elements by distance measurement in a feature space. A generic querying language for the VizIR framework has been developed with concrete instances for particular querying models. For the integration of the VizIR framework into the instance of the Media Pool Accessor and Connectors component, currently the k-nearest neighbor model is used. However, the interfaces between the two frameworks are kept generic to allow for future adaptation of the model and the extension to different models.

The VizIRMediaElementsConnector uses the query object (see Section 8.7.1) for specifying the parameters of a content-based media elements retrieval query. The querying object is passed from the MM4U framework via the VizIRMediaElementsConnector to the VizIR framework. The VizIR framework determines a ranked list of the most suitable media elements according to the given query and returns the query result back to the media connector. The retrieved query result is then converted within the VizIRMediaElementsConnector to the MM4U compliant representation of the media elements. With the integration of the VizIR framework, we enhance the MM4U framework with sophisticated support for content-based retrieval.

8.7.3.4 Multimedia Database METIS

The Digital Memory Engineering group at the Research Studios in Austria [RSA06] developed with the multimedia database METIS a sophisticated storage and management solution for multimedia data [RWP04, KPW04]. The METIS database provides a flexible concept for the definition and management of arbitrary media elements and their meta data. The METIS database is adaptable and extensible to the requirements of a concrete application domain by integrating application-specific plug-ins. This is supported by the possibility for defining domain-specific (complex) media types. For it, the semantic relationship of specific media elements and their meta data can be described to form new, independent multimedia data types. Finally, those domain-specific media types can be bundled up and distributed in form of so-called semantic packs.

With developing a METISMediaElementsConnector for the MM4U framework, the METIS database can be used by concrete personalized multimedia applications as sophisticated multimedia storage and management solution. The METISMediaElementsConnector is flexible in regard of the used semantic packs and the (application-specific) queries that can be executed. For our demonstrator application Pictures4U, a personalized photo albums generator in the web (see Section 10.5), we developed a domain-specific semantic pack for managing photos and photo albums. With this semantic pack, the photos and their descriptions are organized. The personalized

photo album application exploits this information to dynamically generate albums in different presentation formats and for different end devices.

The METIS database has not only been integrated into the MM4U framework via the METISMediaElementsConnector. In addition, also the MM4U component framework itself has been integrated into the METIS multimedia database. Here, the MM4U framework is exploited by the METIS database to provide multimedia content in different presentation formats. This integration direction is provided by the Transformation4U service, described in Section 10.4.

8.7.3.5 Transparent Connection to Multiple Connectors

For our instance of the Media Pool Accessor and Connectors component, we not only implemented connectors to provide access to concrete media storage solutions. We also developed with the TransparentConnectionToMultipleMediaElementsConnectors a media elements connector that integrates multiple arbitrary media elements connectors. This media elements connector provides a transparent access to an arbitrary set of concrete media element connectors like those described above. By this, the TransparentConnectionToMultipleMediaElementsConnectors enables our Media Pool Accessor and Connectors component instance to employ more than one media elements connector at the same time. The transparent connection to the multiple media elements connectors is achieved by a locator object that stores as its parameters an arbitrary set of media connectors' locator objects. Thus the locator object mechanism is applied recursively. When calling the openConnection() service of the TransparentConnectionToMultipleMediaElementsConnectors, it is propagated to all media elements connectors specified in the locator object. This means, that internally a connection is established to all media element connectors that are specified in the locator object. For retrieving media elements by this transparent connection to multiple media elements connectors, the corresponding services getMediumElement(...) and getMediaElements(...) are analogously propagated to the hidden concrete media elements connectors. Here, the concrete media elements connectors are successively called in the order they are specified in the TransparentConnectionToMultipleMediaElementsConnectors' locator object. In the case of the getMediumElement(...) service, this sequential calling stops once a medium element with the requested ID is found. However, with the getMediaElements(...) service, the result sets returned by the concrete media elements connectors are accumulated to a unified result set of the TransparentConnectionToMultipleMediaElementsConnectors. With it however, the ordering of the media elements in the single result sets is lost. A future more sophisticated solution of the transparent connection to multimedia media elements connectors could provide for automatically rearranging the media elements in the unified result set according to the ordering within the single connectors result set. Analogously to the openConnection() service, calling the closeConnection() service of the transparant media elements connector closes all connections of the concrete media elements connectors.

8.7.4 Summary

The Media Pool Accessor and Connectors component is designed for integrating different concrete media elements storage solutions into the MM4U framework. Im-

plementing an instance of the Media Pool Accessor and Connectors component by using the abstract factory pattern and the locator mechanism is only one possibility. In contrast, each of the concrete media element connectors described above could also be developed as distinct instances of the Media Pool Accessor and Connectors component.

Future extensions of our instance of the Media Pool Accessor and Connectors component could be to add digital watermarks to the retrieved media elements on the basis of the user profile information. Also accounting functionality could be provided for billing media content. Such a watermarking and accounting functionality could be build on top of the different concrete connectors for the media elements and meta data. This would enable a transparent usage of the watermarking and accounting functionality, even for media element connectors that do not support it.

8.8 Development of the User Profile Accessor and Connectors Component

The User Profile Accessor and Connectors component provides access to and processing of arbitrary user profile information. It aims at integrating different solutions for storage and processing of user profile information and providing access to that information by a unified interface. The unified user model defined in Section 6.2.2 is used as conceptual basis for specifying the User Profile Accessor and Connectors component and developing our component instance.

8.8.1 Specification of the Component

On basis of the component's group-hot-spot-card and its associated hot-spot-cards, we specified the User Profile Accessor and Connectors component. The provided interface IUserProfileAccessor of the component is shown in Listing 8.2. The services the component provides allows concrete personalized multimedia applications to access, manipulate, and store user profile information and to manipulate, i. e., extend, refine, and consolidate the underlying user model. The provided interface also supports importing and exporting existing user profiles from and to different profile formats.

```
public interface IUserProfileAccessor {
  public void openConnection()
      throws MM4UCannotOpenUserProfileConnectionException;
  public void closeConnection()
      throws MM4UCannotCloseUserProfilesConnectionException;

  public boolean existUserProfile(String profileID);
  public IUserProfile createUserProfile(String profileID)
      throws MM4UCannotCreateUserProfileException;
  public IUserProfile getUserProfile(String profileID)
      throws MM4UUserProfileNotFoundException,
             MM4UInvalidUserProfileException;
  public void setUserProfile(IUserProfile profile)
      throws MM4UCannotStoreUserProfileException;
  public void deleteUserProfile(String profileID)
      throws MM4UCannotDeleteUserProfileException;
```

```
    public IUserModelNode getUserModelRoot();
    public TreeSet getRemovableUserModelNodes();
    public abstract void extendUserModel(
        IUserModelNode nodeToExtend,
        String type,
        String name) throws MM4UCannotExtendUserModelException;
    public void removeUserModelExtension(IUserModelNode extensionNode)
        throws MM4UCannotRemoveUserModelExtensionException;

    public void exportUserProfile(
        String exportFormat,
        String profileID,
        String outputProfileID) throws MM4UCannotExportUserProfileException;
    public IUserProfile importUserProfile(
        String importProfileID,
        String importFormat) throws MM4UCannotImportUserProfileException;
}
```

Listing 8.2: Provided interface of the User Profile Accessor and Connectors component

For using the User Profile Accessor and Connectors component a provided protocol for the IUserProfileAccessor interface is defined. The provided protocol is similar to the protocol of the Media Pool Accessor and Connectors component presented in Section 8.7.2. It is determined by the FSA shown in Figure 8.5. First, a connection to a concrete user profile server is opened with the openConnection() service. Then, the services of the provided interface can be used for getting, setting, creating, and deleting concrete user profiles as well as checking if a specific user profile exists. Here, the services getUserProfile(...), setUserProfile(...), createUserProfile(...), existsUserProfile(...), and deleteUserProfile(...) are provided. In addition, the abstract user model can be extended and refined by application-specific nodes. The root node of the abstract user model can be retrieved by the service getUserModelRoot(...). From this root node, the single associated child nodes of the user model can be recursively accessed. The service extendUserModel(...) is used to actually extend a specific user model node. To obtain a list of all nodes of the abstract user model that are in principal removable, the service getRemovableUserModelNodes() is called. A node in the abstract user model is removable, if the considered node and none of its associated child nodes is employed by a concrete user profile. For importing and exporting concrete user profiles from and to other profile formats, the services importUserProfile(...) and exportUserProfile(...) are applied. Finally, the connection to the concrete user profile server can be closed with the service closeConnection().

8.8.2 Design and Implementation of a Component Instance

The UML class diagram shown in Figure 8.6 depicts the principal design of our instance of the User Profile Accessor and Connectors component. As for the Media Pool Accessor and Connectors component instance, we also use for the design of the User Profile Accessor and Connectors component instance the abstract factory pattern [GHJV04, Mar98] to provide a flexible support for the most different solutions for managing and processing user profile information. For it, also the locator mechanism is employed (see Section 8.7.3).

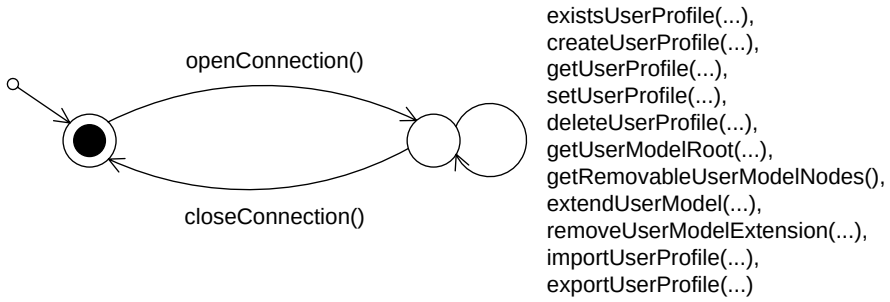


Figure 8.5: Finite state automaton defining the provided protocol of the User Profile Accessor and Connectors component

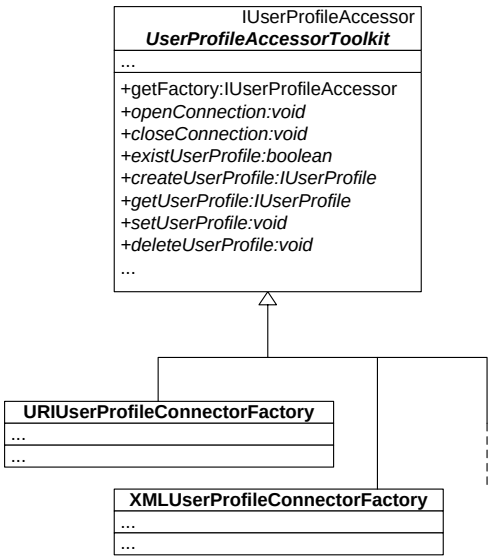


Figure 8.6: UML class diagram of the User Profile Accessor and Connectors component instance

To create a connector for a particular user profile storage solution, a personalized multimedia application calls the getFactory(*<IUserProfileConnectorLocator>*) method of the abstract factory class UserProfileAccessorToolkit with the connector's locator object as parameter. Here, the corresponding factory class is created, e. g., the URIUserProfileConnectorFactory. This factory class creates all necessary internal objects required by the instance of the User Profile Accessor and Connectors component.

As the Media Pool Accessor and Connectors component instance, also the instance of the User Profile Accessor and Connectors component is realized as object-oriented framework. The connectors for the user profile stores are added to our component instance by creating a concrete Factory class of the abstract UserProfileAc-

cessorToolkit. In addition, a corresponding locator class for the user profile connector must be provided. The UserProfileAccessorToolkit class holds the control flow within the User Profile Accessor and Connectors component instance, which calls or propagates the calls to its concrete extensions of the specific user profile stores. As depicted in Figure 8.6, we developed among others the two concrete user profile connectors URIUserProfileConnector and XMLUserProfileConnector for our User Profile Accessor and Connectors component instance. These connectors are described in the following sections.

8.8.2.1 URIUserProfileConnector

The URIUserProfileConnector is able to read and write user profile information via the Internet using the ftp-protocol as well as provides access to this data from the local hard drive. It employs a simple implementation for the unified user model presented in Section 6.2.2. The user model is described as hierarchically ordered key-value pairs. For hierarchically ordering these pairs, the keys can be divided into sub-keys by employing the "."-syntax (dot-syntax). The concrete user profiles are stored in plain text files. A concrete user profile consists of a set of these pairs as the example in Listing 8.3 shows. Within this example, the key User.DemographicalData carries values for the sub-keys such as Name, Contact.Address, and Gender. In addition, the key-value pair model also provides support for list of values. For example, the key User.MentalCharacteristics.Interests determines the users interests. The fact that this key constitutes a list of multiple values is indicated by the $ following the key. The subsequent numbers indicate the elements within the list. As depicted in Listing 8.3, a list node can also have further child nodes such as InterestName and InterestLevel in the given example.

The provided user model on basis of key-value pairs is quite simple but already powerful enough to allow effective pattern-matching queries on the user profiles [CK00]. Due to its simplicity, the URIUserProfileConnector is dedicated to be executed on resource limited devices such as PDAs and cell phones.

The URIUserProfileConnector does not support for importing from and exporting to other formats for user profile information. Thus, only a subset of the FSA defining the component's provided protocol is valid for the URIUserProfileConnector. After opening a connection to the user profile store, the services importUserProfile(...) and exportUserProfile(...) cannot be used. As a consequence, the usage of the User Profile Accessor and Connectors component instance is restricted to the services defined in the FSA excluding those for importing and exporting user profiles (cf. parameterized contracts as introduced in Section 8.6).

```
# User profile
# Wed Apr 19 17:05:28 CEST 2006
User.DemographicalData.Name.FirstName=Peter;Christian
User.DemographicalData.Name.LastName=Schmidt
User.DemographicalData.Name.Salutation=Mr
User.DemographicalData.Contact.Address.Street=Main Street
User.DemographicalData.Contact.Address.HouseNumber=100
User.DemographicalData.Contact.Address.PostalCode=12345
User.DemographicalData.Contact.Address.City=Hanover
User.DemographicalData.Contact.Address.Country=Germany
User.DemographicalData.Contact.EMailList=Peter.C.Schmidt@dummymail.de
User.DemographicalData.Contact.TelephoneNumberList=0123/456789
```

```
User.DemographicalData.Gender=male
User.DemographicalData.Birthplace=Berlin
User.DemographicalData.DateOfBirth=1966-12-31
User.DemographicalData.Nationality=German
User.DemographicalData.Family.MaritalStatus=unmarried
User.DemographicalData.Family.Children=none
User.DemographicalData.OccupationalData.Education=Higher Education
User.PhysicalCharacteristics.CharacteristicsOfTheBody.Weight=90kg
User.PhysicalCharacteristics.CharacteristicsOfTheBody.Height=1.83m
User.PhysicalCharacteristics.CharacteristicsOfTheBody.EyeColor=blue
User.PhysicalCharacteristics.CharacteristicsOfTheBody.HairColor=brown
User.PhysicalCharacteristics.VitalSigns.PulseRate=73
User.PhysicalCharacteristics.VitalSigns.BodyTemperature=36.7
User.PhysicalCharacteristics.PartsOfTheBody.Arms.LeftArm.LeftHand=
User.PhysicalCharacteristics.PartsOfTheBody.Arms.RightArm.RightHand=busy
User.MentalCharacteristics.Interests$0.InterestName=theater
User.MentalCharacteristics.Interests$0.InterestLevel=0.9
User.MentalCharacteristics.Interests$1.InterestName=museum
User.MentalCharacteristics.Interests$1.InterestLevel=0.8
User....
```

Listing 8.3: Extract of a user profile provided by the URIUserProfileConnector

8.8.2.2 XMLUserProfileConnector

A more sophisticated connector to user profile information is the XMLUserProfileConnector. This connector uses XML Schema [W3C05] for modeling the data structure of the unified user model as defined in Section 6.2.2. Consequently, it employs XML as format for storing the user profile information. The XMLUserProfileConnector allows for importing and exporting concrete user profiles from and to other user profile storage solutions and formats by employing XSLT. For example, the text format used by the hierarchical key-value-based representation of user profile information in the previous section can be imported and exported by the XMLUserProfileConnector.

Figure 8.7 depicts an UML-based package diagram of the XMLUserProfileConnector. The central package of this connector is the manager comprising the XMLUserProfileConnectorFactory. This package employs and controls its three subordinate packages for managing the unified user model (model package), the concrete user profiles (repository package), and the import and export of user profiles from and to different standards and existing systems (transformations package). These three packages build the three modules of our XMLUserProfileConnector.

Within the user model module located in the model package, two XML Schema files are used to define the unified user model. The core schema defines the nodes of the abstract user model as they are introduced in Section 6.2.2. In addition, it supports historical aspects of the user profile information. This means that changes of the user profile information can be tracked over time, e. g., when the marital status of a user changes from single to married. Other typically historic-related information that can be modeled and stored are users' purchases (what products have the users when bought) and interaction histories (describing the users behavior on a web site). The core schema is predefined by the XMLUserProfileConnector. It cannot be modified by the concrete personalized applications using the profile connector. To support for managing and storing application-specific extensions and refinements

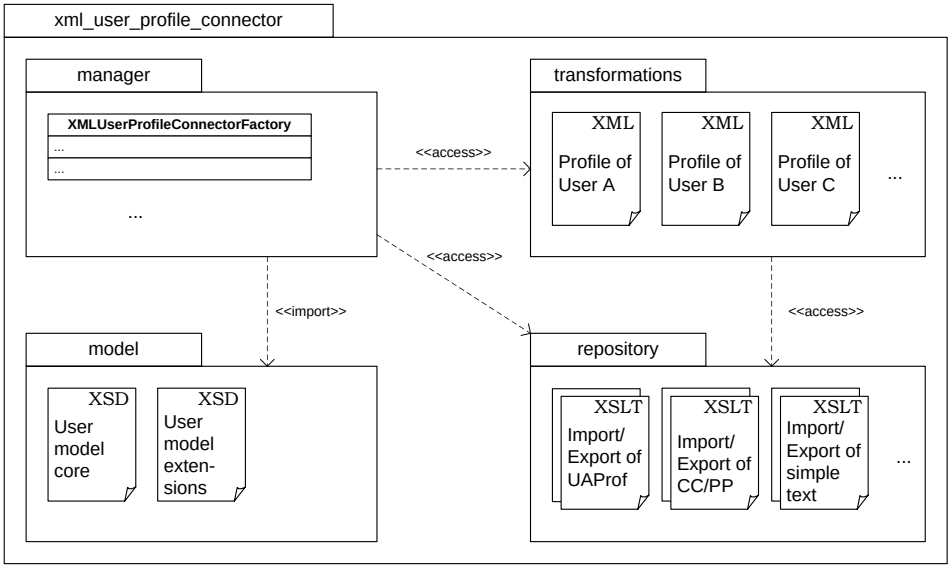


Figure 8.7: Parts of the UML package diagram of the XMLUserProfileConnector connector

of the user model a second schema called extension schema is provided. The extension schema uses the core schema as basis and stores all application-specific extensions and refinements of the unified user model. Once an extension or refinement is defined within the extension schema, it is also available to other personalized applications using the XMLUserProfileConnector.

The repository module of the repository package is the actual store for the user profiles. The concrete user profiles are stored as XML files on the local file system. However, also other formats such as simple text are possible. Here, the Transformations Module needs to be applied for correspondingly exporting and importing the user profiles first. In addition, also other persistence systems can be employed like a XML-based database.

The import and export module located in the transformations package employs XSLT sheets for importing and exporting user profiles in the internal user model from and to different user profile standards and storage solutions. The module allows for defining and applying specific XSLT sheets for importing from and exporting to a specific format or system. The transformation step can also be conducted by defining a sequence of transformation sheets. The XMLUserProfileConnector is employed by the thin-client variant of the tourist guide application Sightseeing4U of our hometown Oldenburg described in Section 10.1.

8.8.3 Summary

The described design and implementation of our User Profile Accessor and Connectors component instance is only one possibility. Like with the Media Pool Accessor and Connectors component, also a completely new instance of the User Profile Ac-

cessor and Connectors component could be developed for each concrete connector. Consequently, in contrast to the given implementation, other instances would not base on the abstract factory pattern and the proposed locator mechanism. These instances of the User Profile Accessor and Connectors component could also base on different technological settings. For example, the user profile information could be managed and stored by employing a relational database connected via JDBC. In addition, it would also be possible to develop a user profile connector that integrates multiple user profile stores and brings them into the MM4U framework. For it, a locator object needs to be developed that carries a set of other user profile locator objects as parameters. With it, the locator mechanism is applied recursively (cf. transparent access to multiple media element connectors presented in Section 8.7.3.5).

8.9 Development of the Multimedia Composition Component

The Multimedia Composition component provides a dynamic composition and assembly of personalized multimedia content. The component allows for coherently arranging media elements in time and space exploiting the internal multimedia content representation model introduced in Section 6.3. This representation model abstracts from the syntax and features of today's multimedia presentation formats.

8.9.1 Underlying Concepts

For creating personalized multimedia content, the abstract multimedia content representation model supports the composition of multimedia content on three different conceptual levels, the basic, complex, and sophisticated multimedia composition functionality. With the basic multimedia composition functionality, a set of application-independent basic composition elements is provided consisting of media elements, basic operators, and projectors (see Section 6.3.3). The complex composition functionality abstracts from the basic composition functionality and allows for encapsulating an arbitrary number of basic composition elements into bigger composition units. These larger composition units are called (parameterized) complex composition operators (see Section 6.3.4). The temporal course of a multimedia presentation defined by a complex operator remains static and cannot be dynamically changed according to different user profile information. Consequently, the Multimedia Composition component allows for sophisticated multimedia composition functionality (see multimedia composition requirement in Section 5.2.1). With sophisticated composition operators additional application logic is exploited to dynamically determine the structure, i. e., temporal course, spatial layout, interaction possibilities, and used media elements of a multimedia presentation (see Section 6.3.5).

8.9.2 Specification of the Component

The analysis of the group-hot-spot-card for the Multimedia Composition component led to one hot-spot-card only. For realizing this hot-spot-card, the doCompose(...) method is introduced in the Multimedia Composition component. It is employed for

realizing the required complex and sophisticated multimedia composition functionality. However, with evolving the MM4U framework it emerged that this hot-spot-card and doCompose(...) method is not sufficient. This was not due to the choice of the hot-spot-card. However, it is due to the flexibility requirements that need to be captured by the multimedia composition and multimedia personalization functionality of the component. Providing for arbitrary complex and sophisticated multimedia composition and multimedia personalization functionality requires to possibly integrate the most different data sources into the Multimedia Composition component. A service like the doCompose(...) method has a static and predefined list of parameters and thus cannot provide for such a flexible functionality. With maturing the MM4U framework, the doCompose(...) service turned out to be the "constructor" of the Multimedia Composition component instances. Here, arbitrary (application-specific) parameters can be defined for the individual sophisticated multimedia composition and personalization functionality. The corresponding provided protocol of the Multimedia Composition component is defined by the FSA shown in Figure 8.8.

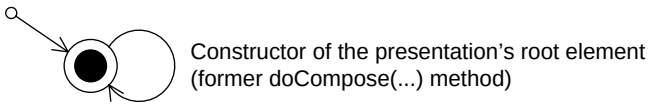


Figure 8.8: Finite state automaton defining the provided protocol of the Multimedia Composition component

8.9.3 Design and Implementation of a Component Instance

For designing and implementing our instance of the Multimedia Composition component, we first designed the internal structure necessary to manage the basic, complex, and sophisticated multimedia composition functionality. This structure is depicted as a UML diagram in Figure 8.9. It consists of several interfaces, defining the relationship between the different basic composition elements, i. e., the projectors, operators, and media elements, as well as the complex and sophisticated operators. The root of all multimedia composition elements is the interface IElement. Consequently, this interface acts as supertype for all composition elements in the Multimedia Composition component (see layer supertype pattern in [FRF+02]). From this supertype, we specialize to projectors and composition variables, using the interfaces IProjector and IVariable. As introduced in Section 6.3.3.2, a composition variable is an abstraction of the media elements and the basic operators. Consequently, there are two interfaces IMedium and IOperator derived from the IVariable interface for the media elements and composition operators. The IOperator interface is further subtyped to distinguish between basic operators using the IBasicOperator interface and IComplexOperator, serving for both complex and sophisticated composition functionality.

For refining the Multimedia Composition component's group-hot-spot-card, the set of basic composition operators, media elements, and projectors are not of further interest. The basic composition elements constitute fixed elements, i. e., atomic units for multimedia composition. However, one requirement to the Multime-

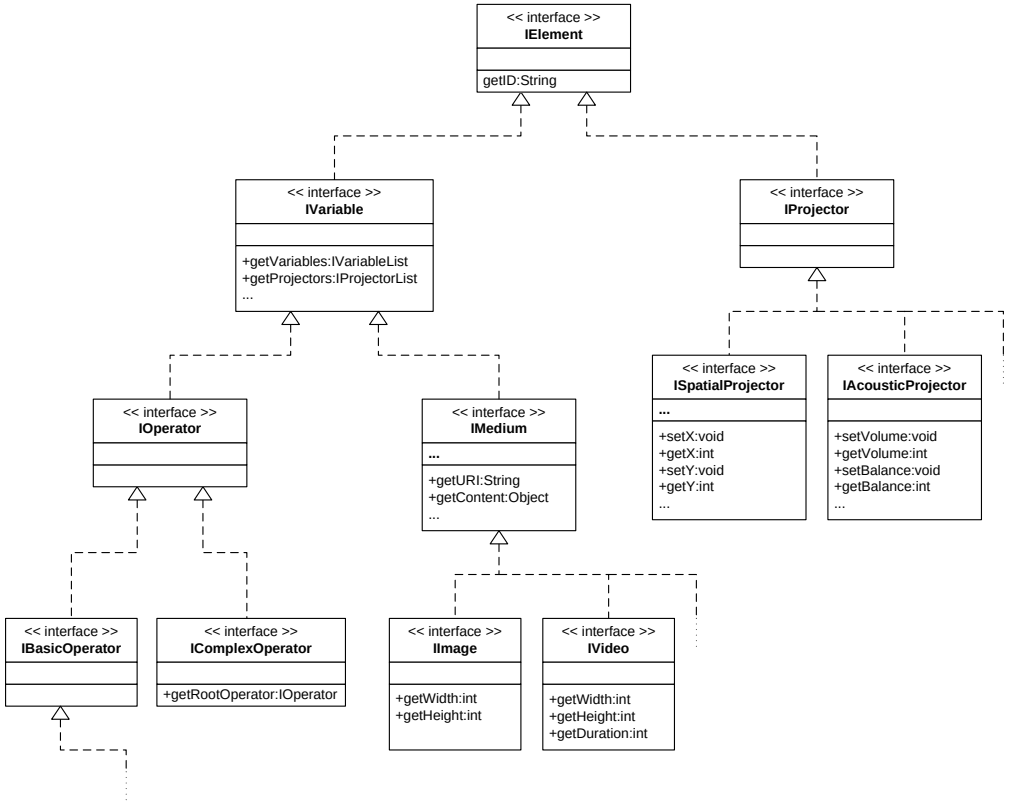


Figure 8.9: UML diagram of the interface relationship of the abstract multimedia representation model

dia Composition component is that it shall be extensible by more complex and application-specific multimedia composition and personalization functionality (see Section 5.2.1). This is provided by the complex composition operators and sophisticated composition operators, respectively. Both kinds of composition operators are realized by the composition interface IComplexOperator.

As described in Section 8.9.2, we initially identified a hook method called doCompose(...) as hot-spot of the Multimedia Composition component. This hot-spot is required for providing complex and application-specific multimedia composition and personalization functionality and is part of the IComplexOperator interface. However, with developing and further refining the Multimedia Composition component this hook method matured to be the constructor of the operators implementing the IComplexOperator interface, i. e., the complex and sophisticated composition operators. Once the hook method is defined in the Multimedia Composition component, its signature is fixed and cannot be modified. Concrete applications would not be able to pass application-specific parameters to their complex operators to realize the required multimedia composition functionality. Thus, with refining the group-

hot-spot-card, we changed the initial hook method to be the constructor of the concrete classes implementing the IComplexOperator interface. The constructor is more flexible, since its signature can be defined individually for each concrete complex operator. In addition, also multiple constructors are possible.

Media Elements The internal representation model's media elements Image, Text, Audio, and Video are realized by the framework interfaces IImage, IText, IAudio, and IVideo. These are subtyped as depicted in Figure 8.9 from the media elements' interface IMedium. For managing and processing media elements in our Multimedia Composition component instance, we use the design pattern proxy [GHJV04]. This means that the media elements, i. e., the respective Java objects, are aimed to merely manage the meta data associated to the media elements. The actual media data of the media elements is loaded into the framework only if necessary (cf. lazy load pattern in [FRF+02]).

Projectors Analogously to the media elements, we derived subtypes of the IProjector interface for the four projectors the internal multimedia composition model offers (see Figure 8.9). These interfaces are ISpatialProjector, IAcousticProjector, ITemporalProjector, and ITypographicProjector. Each projector in our Multimedia Composition component instance is implementing its corresponding interface in a distinct Java class.

Basic Composition Operators Corresponding to the UML specification of the internal multimedia content representation model, we derived three subtypes for the basic composition operators, which are ITemporalOperator, ISelectorOperator, and IInteractionOperator. As depicted in Figure 8.10, we defined the interfaces for the internal model's temporal, selector, and interaction composition operators basing on these three subtypes. Like the media elements and projectors, each of the internal model's temporal, selector, and interaction composition operator implements its corresponding interface in a distinct multimedia composition class (cf. generic content format pattern in [VZ02]).

Application of the Basic Composition Elements As each of the basic multimedia composition elements above is implemented as a distinct class in the concrete instance of the Multimedia Composition component, an application developer uses them as black-box and creates as much instances of these classes as needed to compose the personalized multimedia content. With creating these instances, the constructors of the basic composition elements' classes ensure that any internal objects required for the multimedia composition are created. It is the responsibility of the constructors and methods of these classes to actually create all necessary objects and references for composing the multimedia content. Consequently, the abstract multimedia content representation model is also called an object-oriented representation of the personalized multimedia content.

Complex Composition Operators For the development of complex multimedia composition functionality, the composition interface IComplexOperator is implemented by

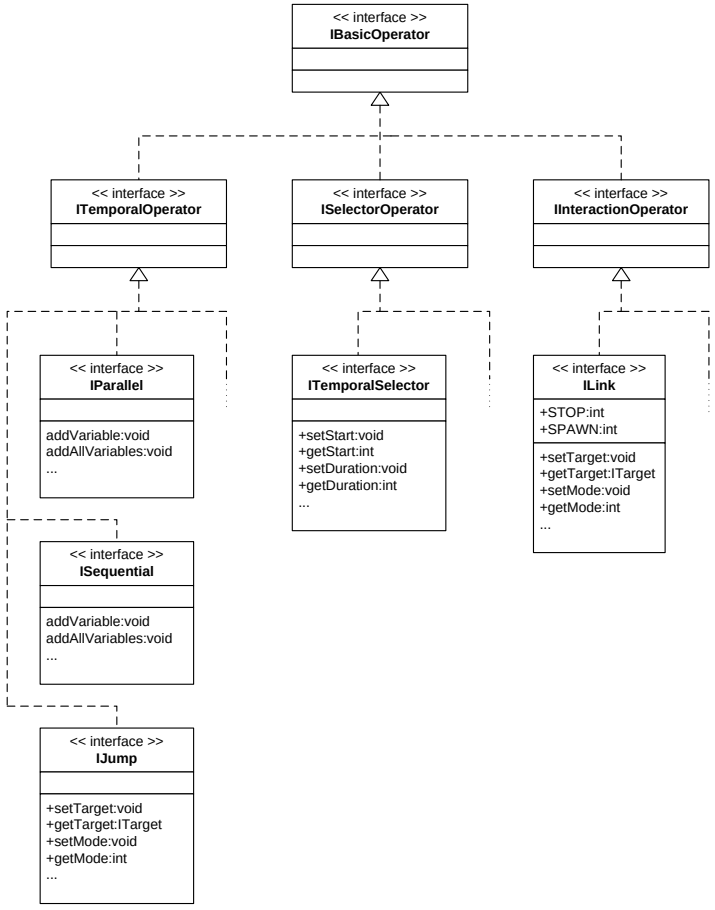


Figure 8.10: UML diagram of the interface hierarchy of the basic composition operators

concrete complex composition operators. In addition, also an abstract implementation of this interface is provided by the Multimedia Composition instance with the AbstractComplexOperator class. For application developers it is easier to use the AbstractComplexOperator class since it already provides some standard functionality the IComplexOperator interface defines. Like the basic composition elements, the complex composition operators are used by creating objects (i. e., instances) of the corresponding concrete complex composition operator classes.

To provide access to the multimedia presentation generated by a complex operator, the interface IComplexOperator defines the method getRootOperator(). This method returns the root element of the generated presentation. Starting with this root element, all other multimedia composition elements in the multimedia representation tree can be accessed recursively. The getRootOperator() method is of importance for the actual transformation of the internal representation model into

the different concrete multimedia presentation formats by the Presentation Format Generators component presented in Section 8.10.

Sophisticated Composition Operators Despite the different multimedia composition characteristics of complex operators and sophisticated operators, both base on implementing the IComplexOperator interface. The reason for this lies in the fact that for the generated multimedia presentation it is insignificant if the structure of the personalized multimedia presentation is predefined and hard-wired within a (parameterized) complex composition operator or if the structure is dynamically determined by a sophisticated operator during runtime and depending on the actual user profile information. This is also independent of what kind of user profile information or other additional parameter and data sources are used for fulfilling the composition and personalization task (cf. Section 6.3.4 and Section 6.3.5). Complex operators and sophisticated operators implement the IComplexOperator interface providing the formerly defined doCompose(...) service, which matured to the constructor of these operators. Consequently, each concrete complex and sophisticated composition operator can be considered as an instance of the Multimedia Composition component.

Covariance and Contravariance in the Multimedia Content Representation Model It is important to note that with subtyping the multimedia composition elements, i. e., the projectors, media elements, and composition operators as depicted in Figures 8.9 and 8.10 only new services are added to the defined subtypes. This means, that there is no descendant hiding [Mey97, 627ff.], i. e., existing services provided by a supertype are not hided in the subtypes. In addition, there is no redefinition of the existing services' parameters or return values in the subtyped composition elements. By this, we evade the problems that can occur with covariance and contravariance. For a detailed discussion of these issues see [Mey97, 621ff.].

8.9.4 Summary

The Multimedia Composition component provides the basic functionality required for the composition of multimedia content. In addition, it can be extended by complex and application-specific multimedia composition and personalization functionality. In future work, the set of basic multimedia composition elements could be extended to provide support for more fancy composition operators, e. g., transition effects like in SMIL or projectors for different visual effects such as blurring in SVG. Besides this, the spatial arrangement of the composition is conducted so far in a 2D space. In addition, a z-order value of the visual media elements can be specified to reach a 2.5D space. For providing support for a spatial composition in a real 3D space, an appropriate abstract description format for interactive 3D scenes and worlds would be needed. With the CONTIGRA project (COmponent-orieNted Threedimensional Interactive GRaphical Applications), research is conducted that aims at providing such an abstract description format for interactive 3D scenes and applications [DHM02]. This XML-based format bases on and extends the Extensible 3D (X3D) [Web06] format and is aimed to be translated into different (proprietary) 3D technologies. Within the context of the CONTIGRA project, also research in

regard of developing an abstract audio format for 3D audio playback is conducted [HDM03]. Here, the aim is to provide a XML-based abstract specification language for 3D audio scenes, which is independent of specific sound APIs and sound reproduction systems.

In addition, the Multimedia Composition component could be extended to support typical form elements as they are defined in HTML and XHTML. Examples for such form-elements are check boxes, combo boxes, radio buttons, selection fields, text input fields, and submit buttons. Such an extension of the Multimedia Composition component instance and abstract multimedia content representation model by form elements has been conducted for our demonstrator application Manual4U providing for multimedia instruction manuals on mobile devices (see Section 10.3).

8.10 Development of the Presentation Format Generators Component

This section describes the development of the Presentation Format Generators component for transforming the multimedia content in the internal representation model provided by the Multimedia Composition component to the features and syntax of the different concrete (mobile) multimedia presentation formats. With it, the Presentation Format Generators component paves the last mile for multi-channel multimedia presentation generation, i. e., the delivery of multimedia content to a wide range of concrete presentation formats and end devices, including Desktop PCs, laptops, and other mobile devices like PDAs and cell phones.

8.10.1 Underlying Concepts

The Presentation Format Generators component's task is to adapt the characteristics and features of the internal multimedia content representation model provided by the Multimedia Composition component in regard of the used time model, spatial layout, and interaction possibilities to the particular characteristics and syntax of the concrete presentation formats. For it, the Presentation Format Generators component provides application-independent transformation algorithms (see Sections 6.4.1 to 6.4.4) in conjunction with distinct sets of transformation rules for mapping the internal representation model to the syntax and features of the different targeted presentation formats (see Sections 6.4.5 to 6.4.7).

For example, the temporal course of the internal multimedia content representation model is realized by nested local timelines (see Section 6.3.1.1). Such a time model with local timelines is supported by the presentation format SMIL and its derivates. However, SVG and Flash provide for a single global timeline only. Here, the Presentation Format Generators component transforms the nested local timeline to a global timeline by applying the algorithm presented in Section 6.4.1.

The spatial layout of our internal representation model is realized by a hierarchical model that supports the positioning of media elements in relation to other media elements (see Section 6.3.1.2). This relative positioning is supported by most of today's presentation formats, e. g., SMIL, SVG, and MPEG-4's XMT-Ω. However, SMIL BLP, 3GPP SMIL, and Flash only allow an absolute positioning of visual media elements in regard of the presentation's origin. In this case, the Presentation

Format Generators component transforms the hierarchically organized spatial layout of the internal representation model to a spatial layout of absolute positioning by applying the algorithm described in Section 6.4.2.

For mapping the basic multimedia composition elements to the syntax of the targeted presentation formats, we need to consider the syntactic structure how the targeted presentation formats define their multimedia content. For defining the multimedia content, the concrete presentation formats analyzed in Section 6.3.1 provide either for two distinct parts or define the multimedia content in one large section only. When defining the content in two distinct parts, typically a header and a body of a presentation is provided. Within the header, media elements can be defined, e. g., with Flash, or the spatial and acoustic layout of the presentation is determined, e. g., in the SMIL family. Within the presentation's body, typically the temporal course of the multimedia presentation is determined such as in the SMIL family. When providing only one section for specifying the multimedia presentation, the spatial and acoustic layout are defined together with the temporal course of the multimedia content. This is provided, e. g., by SVG and its profiles.

8.10.2 Specification of the Component

For the specification of the Presentation Format Generators component, we refined the corresponding group-hot-spot-card and identified a set of hot-spot-cards on the granularity of single methods. The central hot-spot for using the Presentation Format Generators component is provided by the service doTransform(...). This service allows for transforming the multimedia content in the internal representation model to the syntax and features of the targeted presentation formats. It is defined in the provided interface IGenerator of the component, which is shown in Listing 8.4.

```
public interface IGenerator {
  public IMultimediaPresentation doTransform(
          IVariable rootVariable,
          String title,
          IUserProfile profile) throws MM4UGeneratorException;

  public IMultimediaPresentation doTransform(
          IVariable rootVariable,
          String title,
          int width,
          int height,
          IUserProfile profile) throws MM4UGeneratorException;
}
```

Listing 8.4: Provided interface of the Presentation Format Generators component

The doTransform(...) service comes in two variants. Both variants require as first parameter an instance of IVariable as input, which is the root node of the multimedia composition tree assembled by the Multimedia Composition component. However, the first variant determines the presentation's spatial dimensions automatically. Its further parameters are the title of the multimedia presentation and the user's profile information. The latter is exploited for some additional personalization tasks in the Presentation Format Generators component, e. g., setting the background color

of the presentation the user's preference. With the second variant of the doTransform(...) service, the application developers can manually determine the width and height of the presentation.

In both cases, the return value of the doTransform(...) service is the personalized multimedia presentation in a target presentation format. This target presentation format can be either a XML-based data structure such as SMIL and SVG but also a binary multimedia presentation stream like Flash and MPEG-4. For managing the target presentations in both a XML-based structure and binary format, an appropriate abstraction is used with the IMultimediaPresentation interface.

As the provided interface of the Presentation Format Generators component comprises only the doTransform(...) service, the corresponding provided protocol is quite simple. It is defined by the FSA depicted in Figure 8.11.

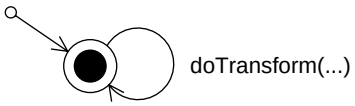


Figure 8.11: Finite state automaton defining the provided protocol of the Presentation Format Generators component

8.10.3 Design and Implementation of a Component Instance

For developing a concrete instance of the Presentation Format Generators component, we refined its group-hot-spot-card and a set of hot-spot-cards were identified that describe the internal realization of the component. For designing the instance of our Presentation Format Generators component, we applied the design pattern abstract factory [GHJV04, Mar98]. The abstract factory is provided by the GeneratorToolkit class, which also realizes the component's provided interface IGenerator.

An application developer creates an instance of the Presentation Format Generators component by calling the GeneratorToolkit's getFactory(*<targetFormat>*) method. The integer parameter targetFormat determines the targeted presentation format. Then, one of the two variants of the doTransform(...) service of the Presentation Format Generators component can be called to actually initiate and conduct the transformation process.

As shown in Figure 8.12, the abstract factory has two products, one for generating the temporal course and one for generating the layout of the multimedia presentation (cf. header and body in Section 8.10.1). The abstract implementation of these two products are the hook classes AbstractContentGenerator and AbstractLayoutGenerator. They provide the algorithms for adapting the internal representation model's characteristics to the features of the targeted presentation formats. The products of the abstract factory are the concrete instances of the AbstractContentGenerator and AbstractLayoutGenerator classes, each derived for a specific presentation format. These concrete transformer classes actually map the abstract document tree's composition elements to the syntax of the concrete target formats (cf. content format builder pattern in [VZ02]). Hence, they realize the transformation rules introduced in Section 6.4. The hook class AbstractLayoutGenerator is only necessary for

multimedia presentation formats that define the layout in a header section separate to the actual multimedia content, e. g., SMIL and MPEG4's XMT-Ω within the <header> tag. In cases where the hook class AbstractLayoutGenerator is not required, the corresponding createLayoutGenerator(...) method in the concrete factory classes returns null.

Examples of concrete products for our Presentation Format Generators component instance are the concrete classes derived from the AbstractContentGenerator and AbstractLayoutGenerator classes for the multimedia presentation formats SVG, SMIL, Flash, and XMT-Ω. For the presentation formats SVG and SMIL, there exists an abstract specialization with the classes AbstractSMILContentGenerator and AbstractSVGContentGenerator, respectively. In addition, for SMIL also a corresponding class AbstractSMILLayoutGenerator is defined. As SVG does not define the layout in a separate section of the multimedia presentation description, there is no layout generator class for this format. From these abstract classes, concrete generator classes for content and layout are derived for the profiles and variants of SMIL and SVG with their individual characteristics and features. For example, the classes SMIL_BLPContentGenerator and SMIL_BLPLayoutGenerator for the Basic Language Profile of SMIL are derived. In addition, for the TinyLine SVG player a specific content generator class TinyLinePlayerSVGContentGenerator is developed. Since the structure and syntax of MPEG-4's representation format XMT-Ω bases on SMIL, its generator class `XMTOmegaContentGenerator` is also derived from the AbstractSMILContentGenerator class. However, the XMTOmegaLayoutGenerator is not derived from the AbstractSMILLayoutGenerator class as one could expect, because the definition of the layout in XMT-Ω is very different to SMIL, despite its very high similarity to the SMIL syntax. As the binary Flash format and its XML-based representation FML do not base on any of these presentation formats, the corresponding FlashContentGenerator class is directly derived from the AbstractContentGenerator class. Although, the Flash format provides a <header> section there is no concrete class derived from the AbstractLayoutGenerator class for this format. The <header> section in Flash is merely targeted for defining the media elements used within the multimedia presentation rather than for determining the layout of the Flash presentation.

Each hot-spot-card identified for realizing the Presentation Format Generators component instance is targeted at transforming one of the basic composition elements of our internal multimedia content representation model into the syntax of the different concrete multimedia presentation formats. The hook methods of the identified hot-spot-cards are arranged into the two hook classes AbstractContentGenerator and AbstractLayoutGenerator as depicted in Figure 8.13. Examples of these hook methods are image(...), text(...), audio(...), and video(...) to transform the media elements. For the basic composition operators, hook methods such as parallel(...), sequential(...), and temporalSelector(...) are defined. The projectors are transformed by integrating hook methods such as spatialProjector(...) and acousticProjector(...) into the framework component.

Additional hook methods for transforming the complex and sophisticated composition operators are not necessary, since the corresponding abstract transformation algorithm automatically detects them and decomposes them into their basic bricks, i. e., into media elements, basic composition operators, and projectors. For it, the transformation algorithm calls the method getRootOperator() of the complex and sophisticated composition operators' classes to retrieve their root operator (see Sec-

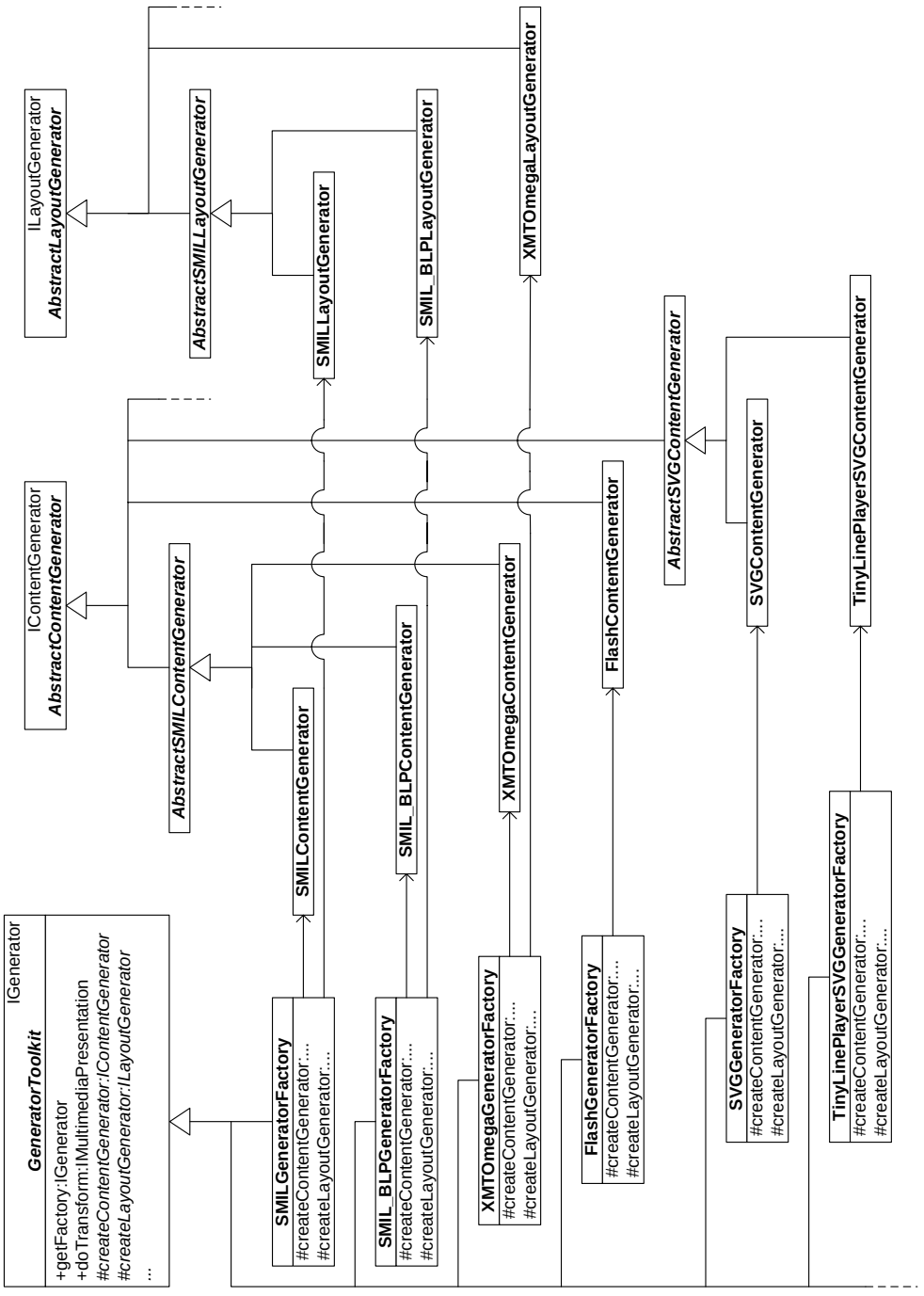


Figure 8.12: Design of the Presentation Format Generators component instance

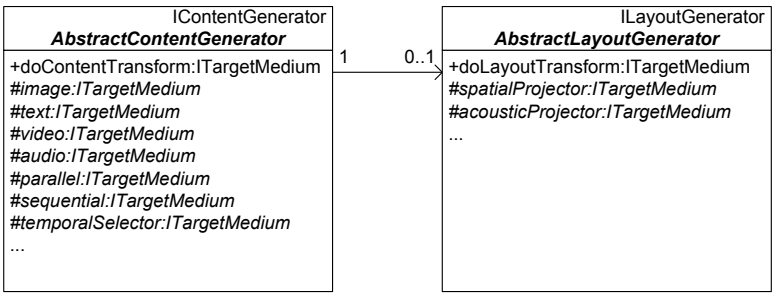


Figure 8.13: UML class diagram of the hook methods of the Presentation Format Generators component

tion 8.9.3). Starting with this root operator, the transformation algorithm traverses the multimedia document tree generated by a complex or sophisticated operator and applies the hook methods introduced above for each basic multimedia composition element.

With the generic generator classes AbstractContentGenerator and AbstractLayoutGenerator the abstract algorithm for traversing the multimedia content in the abstract representation model as well as the abstract algorithms for adapting the temporal model and spatial model to the features of the target format need to be implemented only once. By creating concrete instances of these classes, the Presentation Format Generators component instance can be adapted and applied for the transformation of multimedia content into the syntax of the different presentation formats. For it, besides the concrete subclasses of the AbstractContentGenerator and AbstractLayoutGenerator classes only a concrete presentation format generator factory class needs to be developed for each target format. As the transformation algorithms provided by the Presentation Format Generators component are application-independent, developing such concrete classes for a specific presentation format enables the MM4U framework to provide support for this presentation format in arbitrary personalized multimedia applications.

The design and implementation of the presented instance of the Presentation Format Generators component is conducted as traditional object-oriented framework. The control flow within the component instance is held by the AbstractContentGenerator and AbstractLayoutGenerator classes and not by their concrete extensions for the particular presentation formats. The hook methods such as image(...), text(...), parallel(...), sequential(...), spatialProjector(...), and acousticProjector(...) defined in the AbstractContentGenerator and AbstractLayoutGenerator classes are implemented in the concrete content and layout generator classes. These implementations of the hook methods are called by the AbstractContentGenerator and AbstractLayoutGenerator classes following the call-back-principle.

8.10.4 Supported Multimedia Presentation Formats

For our instance of the Presentation Format Generators component, we implemented concrete generator classes for the presentation formats SMIL 2.0, SMIL 2.0 in the Basic Language Profile for mobile devices, SVG 1.2, Mobile SVG 1.2 con-

taining SVG Tiny and SVG Basic, Flash, and MPEG-4's XMT-Ω. We also developed generator classes for HTML [RLHJ99], XHTML [XHT05], and XHTML Basic [BIM+00] targeted at PDAs and cell phones. The support of the mobile presentation formats is part of the mobile version of our MM4U framework, which is presented in Section 8.16.

The implementation for generating the XML-based presentation formats such as SMIL and SVG is conducted without applying additional programming libraries but by the means provided by the Presentation Format Generators component instance only. However, for implementing the generators of the binary presentation formats additional Java libraries are employed: The transformation of the FML to the binary Flash format is supported by an enhanced version of the Java SWF toolkit [Mai03]. For mapping multimedia presentations in XMT-Ω to the binary MPEG-4 format, we employ IBM's toolkit for MPEG-4 [IBM06b].

As there exist multimedia players that do not support some of the presentation format's tags although defined in the specification, we had to develop further multimedia player specific generator classes (see Section 8.10.3). For example, the Tinyline SVG player [Gir06] does not support the <set> tag of SVG. Consequently, we could not use the <set> tag to dynamically add or remove media elements from the presentation. Here, we had to develop a player specific presentation format generator that pursues a more complicated solution using four <animate> tags instead of one <set> tag (see Section 6.4.6). In addition, not all multimedia players do support the playback of media elements with infinite duration, e. g., the PocketSMIL player [INR02] for PDAs and image elements. So the infinite presentation duration of discrete media elements is shortened to a finite value in the specific generator classes for the PocketSMIL player. However, this value is set to a very long time period of 100 hours such that it gives the end users the impression to be infinite.

The transformation algorithms and AbstractContentGenerator and AbstractLayoutGenerator classes are flexible in regard of any possible combinations of the target format's concrete characteristics for the central multimedia modeling aspects of time, space, and interaction. As we can expect that future multimedia presentation standards will also have to provide means for the central multimedia modeling aspects captured by our internal multimedia content representation model, we are prepared to support new versions of the presentation formats or even new formats. Thus, our instance of the Presentation Format Generators component is flexible such that it can be configured to the characteristics and features of the individual concrete presentation formats.

8.10.5 Summary

The Presentation Format Generators component is not only embedded into our MM4U framework for personalized multimedia content creation but can also be seen as a service provider for systems and applications that create multimedia content and need to deliver this content over the last mile to the individual user's end device setting. Consequently, we set up a Transformation4U service in the Internet, where users can send us a multimedia document in a XML representation of our internal multimedia content representation model (see Appendix C), specify the targeted output format, and receive the multimedia presentation in the final format. This web service and its application is presented in Section 10.4.

8.11 Development of the Multimedia Presentation Component

The personalized multimedia content transformed by the Presentation Format Generators component is finally rendered and displayed by the Multimedia Presentation component on the user's end device for actual consumption. As the Presentation Format Generators component provides multimedia presentations in standardized formats, the Multimedia Presentation component can fulfill the requirement to use and rely on existing multimedia presentation software (see Section 5.2.2). By this, application developers are relieved from the burden of developing their own multimedia player. The existing multimedia player software is typically provided and deployed as individually and independently executable unit. Thus, employing these multimedia players can be regarded as using a multimedia player component within the Multimedia Presentation component, i. e., nesting of software components.

The coupling between the Multimedia Presentation component and the used multimedia player software can be tight or loosely. When the multimedia player software is fully integrated into the Multimedia Presentation component instance, a tight coupling exists. However, also remote multimedia player software can be used that is only loosely coupled with the Multimedia Presentation component, e. g., in the case of a client/server-architecture where the player software is installed on a remote (mobile) client and the MM4U framework resides on a server. Here, the multimedia player software connects as client via a HTTP-connection to the Multimedia Presentation component of the personalized multimedia application.

8.11.1 Specification of the Component

For using the Multimedia Presentation component, the interface IMultimediaPlayer as shown in Listing 8.5 is provided. The interface provides services for controlling the multimedia player software such as starting the playback of a presentation, pausing and resuming it, and stopping the multimedia presentation.

The corresponding provided protocol of the Multimedia Presentation component is defined by the FSA as shown in Figure 8.14. First, the playback of a multimedia presentation is started. This can be conducted in two ways: As the personalized multimedia presentations generated by the Presentation Format Generators component are hold in an object implementing the IMultimediaPresentation interface, it can be directly consumed by the Multimedia Presentation component's play(*<IMultimediaPresentation>*) service. However, the Multimedia Presentation component can also playback a multimedia presentation that is stored on the local hard drive or available on the Internet. Here, the play(*<URI>*) service is applied. This enables the Multimedia Presentation component to be used in other contexts than the MM4U framework. The playback of a multimedia presentation can be arbitrary times paused and resumed by using the appropriate services pause() and resume(). This can be provided, e. g., by the graphical user interface of a personalized multimedia application such as our thick-client variant of the Sightseeing4U application presented in Section 10.1.4. To quit the playback of a presentation, the stop() service is called. Here again, a concrete personalized multimedia application could offer the users a stop button.

```
public interface IMultimediaPlayer {
  public void play(IMultimediaPresentation presentation)
        throws MM4UCannotDisplayMultimediaPresentationException;
  public void play(URI uri)
        throws MM4UCannotDisplayMultimediaPresentationException;
  public void pause();
  public void resume();
  public void stop();
}
```

Listing 8.5: Provided interface of the Multimedia Presentation component for rendering multimedia presentations

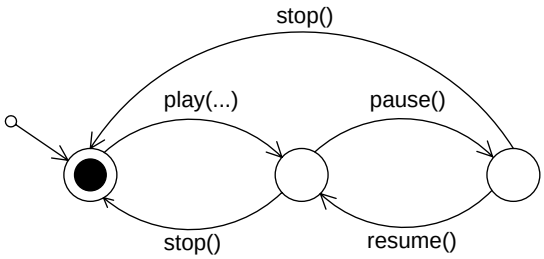


Figure 8.14: Finite state automaton defining the provided protocol of the Multimedia Presentation component

8.11.2 Design and Implementation of a Component Instance

In regard of integrating existing multimedia player software into the Multimedia Presentation component as introduced above, we developed an instance of the component based again on the design pattern abstract factory and the locator mechanism (see Sections 8.7.3 and 8.8.2). The principal design of our instance of the Multimedia Presentation component is shown in Figure 8.15. For this instance, we integrated Java-based multimedia player software. For example, for the thick-client version of our mobile tourist guide application Sightseeing4U (see Section 10.1.4), we exploit TinyLine's SVG player [Gir06]. In addition, we also integrated IBM's MPEG-4 player [IBM06a] and a simple HTML viewer [GC96]. These multimedia players are tightly coupled with the Multimedia Presentation component instance.

Regarding the usage of remote multimedia players on the user's (mobile) end device, we employ among others the RealPlayer [Rea06], Adobe's Flash Plugin [Ado06c] for the various web browsers and end devices, Adobe's SVG viewer plugin [Ado06a] for Microsoft's Internet Explorer, as well as the PocketSVG player [CSI05] and PocketSMIL player [INR02] for PDAs. This remotely connected multimedia player software is used for different demonstrator applications we developed employing the MM4U framework. For example, the thin-client variant of our personalized tourist guide Sightseeing4U presented in Section 10.1.3.

To connect the remote multimedia player software with our instance of the Multimedia Presentation component, the HttpServletPlayerFactory class depicted in

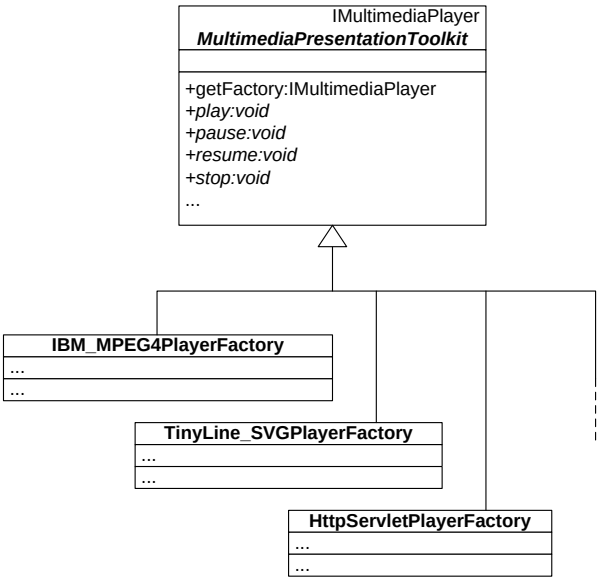


Figure 8.15: UML class diagram of the Multimedia Presentation component instance

Figure 8.15 is used. The HttpServletPlayer acts as wrapper for personalized multimedia applications to transmit their multimedia presentations in the final formats to the remote multimedia player software via a Java servlet. This results in a loose coupling of the remote multimedia player software and the Multimedia Presentation component instance. The HttpServletPlayerFactory is typically used by personalized multimedia applications basing on a client/server-architecture. On the (mobile) client resides the used multimedia player software. This player software is connected to the Multimedia Presentation component instance in form of the HttpServletPlayerFactory on the server side.

Multimedia player software that is loosely coupled via the HttpServletPlayerFactory usually does not support the full provided protocol specified by the FSA in Figure 8.14. Typically, the services pause(), resume(), and stop() for externally triggering to pause, resume, and stop the playback of a multimedia presentation are not supported. Consequently, the usage of the Multimedia Presentation component instance is restricted to a corresponding subset of the FSA (cf. parameterized contracts as introduced in Section 8.6).

8.11.3 Summary

The presented instance of the Multimedia Presentation component provides for a tight and loose coupling of different existing multimedia player software. These existing player software can be considered as subcomponent of the Multimedia Presentation component instance. In regard of user interaction, the Multimedia Presentation component is besides the pause(), resume(), and stop() services not concerned with the interaction of the user and the personalized multimedia presentation. For

example, when a user clicks on an interactive link within the multimedia presentation, this click is not passed back to the Multimedia Presentation component. The event is directly propagated to the concrete personalized multimedia application.

In regard of a future extension of the MM4U component framework towards the evaluation of the personalization quality by means of relevance feedback, an appropriate event handling mechanism could be integrated into the Multimedia Presentation component. Thus, the Multimedia Presentation component would not only be used to playback the multimedia presentations but would also serve as a unified solution to receive incoming events from the multimedia player software activated by mouse-clicks of the users within the multimedia presentation. An example of a personalized multimedia application employing such a relevance feedback functionality receiving events from the employed multimedia player is the personalized sports news ticker Sports4U presented in Section 10.2.

8.12 Development of the MM4U Component

In the previous Sections 8.7 to 8.11, we showed how component technology can be used to provide modular support for the different tasks of the general multimedia personalization process. Implementing these different tasks as distinct software components, it is now a logical consequence to design and implement the embracing MM4U component framework itself as a software component (see also [OB02, p. 219]). Consequently, we created a sixth group-hot-spot-card for a MM4U component that captures the flexibility requirements of the entire MM4U component framework. This group-hot-spot-card describes the flexibility requirements of the single framework components on an even higher abstraction level. In addition, this component defines the order in which the single components of the MM4U framework as presented in Sections 8.7 to 8.11 are to be used [SB05a].

8.12.1 Specification of the Component

The specification of the MM4U component by the corresponding group-hot-spot-card merges the five components identified and specified for the MM4U framework into a unifying interface. This interface is called IPersonalizedMultimediaApplication and specifies the services an application developer needs to implement a concrete personalized multimedia application on the basis of the MM4U component framework. Thus, the IPersonalizedMultimediaApplication interface constitutes the provided interface of the MM4U component. The details of this interface are shown in Listing 8.6.

```
public interface IPersonalizedMultimediaApplication {
  public boolean existUserProfile(IUserProfileAccessor profileConnector,
      String profileID);
  public IUserProfile createUserProfile(IUserProfileAccessor profileConnector,
      String profileID) throws MM4UCannotCreateUserProfileException;
  public IUserProfile getUserProfile(IUserProfileAccessor profileConnector,
      String myUserProfileID, IPropertyList temporaryAdditionalProperties)
      throws MM4UUserProfileNotFoundException,
          MM4UInvalidUserProfileException;
  public IUserProfile updateUserProfileInformation( IUserProfile userProfile );
  public void setUserProfile(IUserProfileAccessor profileConnector,
      IUserProfile myUserProfile)
```

```
        throws MM4UCannotStoreUserProfileException;
    public void deleteUserProfile(IUserProfileAccessor profileConnector,
        String profileID) throws MM4UCannotDeleteUserProfileException;

    public IVariable doCompose(IUserProfile myUserProfile,
        IMediaElementsAccessor mediaAccessor)
        throws MM4UCompositionException,
               MM4UMediaElementsConnectorException;

    public IMultimediaPresentation doTransform(IGenerator myGenerator,
        IVariable rootVariable, int myPresentationWidth,
        int myPresentationHeight, IUserProfile myUserProfile)
        throws MM4UGeneratorException;

    public void doPresent(IMultimediaPlayer myMultimediaPlayer,
        IMultimediaPresentation personalizedMultimediaPresentation)
        throws MM4UPlayerException;
}
```

Listing 8.6: Provided interface for using the MM4U component framework

The provided protocol of the MM4U component is defined by the FSA depicted in Figure 8.16. First, it can be checked by calling the existsUserProfile(...) service whether a specific user profile exists. If it does not exist, the specified user profile can be created by using the createUserProfile(...) service. Otherwise, the profile is retrieved by calling the getProfile(...) service. Then, the current user profile can be updated with the updateUserProfile(...) service, e. g., by applying an appropriate relevance feedback component, and saved back into the user profile store. Subsequently, the personalized multimedia content is created with the doCompose(...) service. Following the composition, the content is transformed into the target format with the doTransform(...) service and presented on the user's end device by applying the doPresent(...) service.

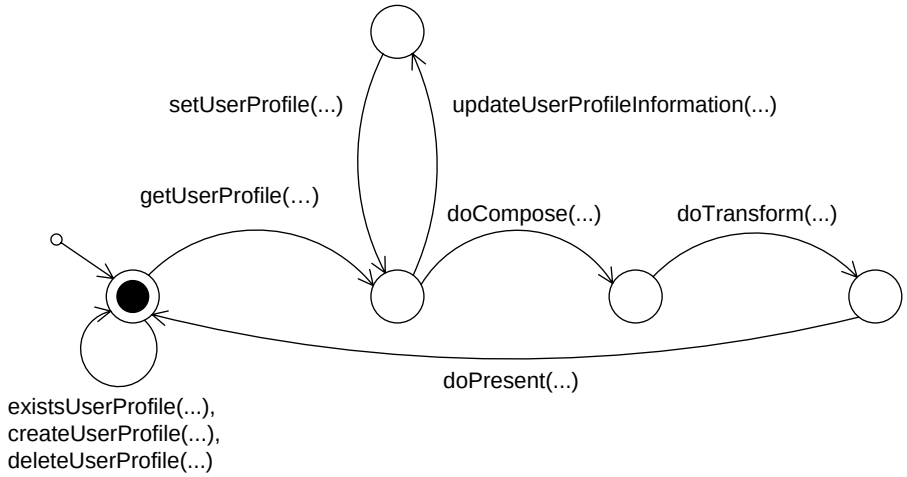


Figure 8.16: Finite state automaton defining the provided protocol of the MM4U component

8.12.2 Design and Implementation of a Component Instance

To develop an instance of the MM4U component, the provided interface IPersonalizedMultimediaApplication needs to be implemented. This is done by the demonstrator applications using our MM4U framework in Section 10. Consequently, these concrete personalized multimedia applications are concrete instances of the MM4U component. Besides the MM4U component framework, arbitrary other application-specific software components can be employed by concrete personalized multimedia applications. For example, a sightseeing application could employ an online ticket ordering component allowing its users to instantly order and pay concert tickets, museum entry fees, and others.

Application developers can execute their concrete personalized multimedia application by using the MM4U component's MM4UApplicationRunner class. This class captures the sequence of calling the services that are defined in the IPersonalizedMultimediaApplication interface. The application runner class implements one possible order of using the MM4U component's services according to the provided protocol depicted in Figure 8.16. With it, the MM4UApplicationRunner class specifies when and how the framework's components interact. This order is depicted by the sequence diagram in Figure 8.17.

8.12.3 Summary

With the MM4U component, we provide a software component that encapsulates the different components of the MM4U framework as described in Sections 8.7 to 8.11. By defining the order of how to use the MM4U component's services by the FSA in Figure 8.16, implicitly also the order of employing the encapsulated framework components is defined. The presented MM4UApplicationRunner class determines one possible path through the FSA and provides application developers support for executing their concrete personalized multimedia applications.

8.13 Other Design Issues of the MM4U Framework

In the previous sections, we presented the development of the single components of the MM4U framework and the development of the MM4U component itself. Besides the specific design and implementation issues of these components, we were faced with some more general design issues during the development of the MM4U component framework. These include considering performance aspects and a common exception concept for the component framework.

8.13.1 Performance Issues

Performance issues had to be taken into account for the design and implementation of the MM4U component framework, since the framework shall be employable on web servers with high load. In addition, the framework shall be applicable and executable on mobile devices with limited resources, i. e., with limited memory and computation power in comparison to a standard Desktop PC, such as PDAs and cell phones (see requirements specified in Section 5.2).

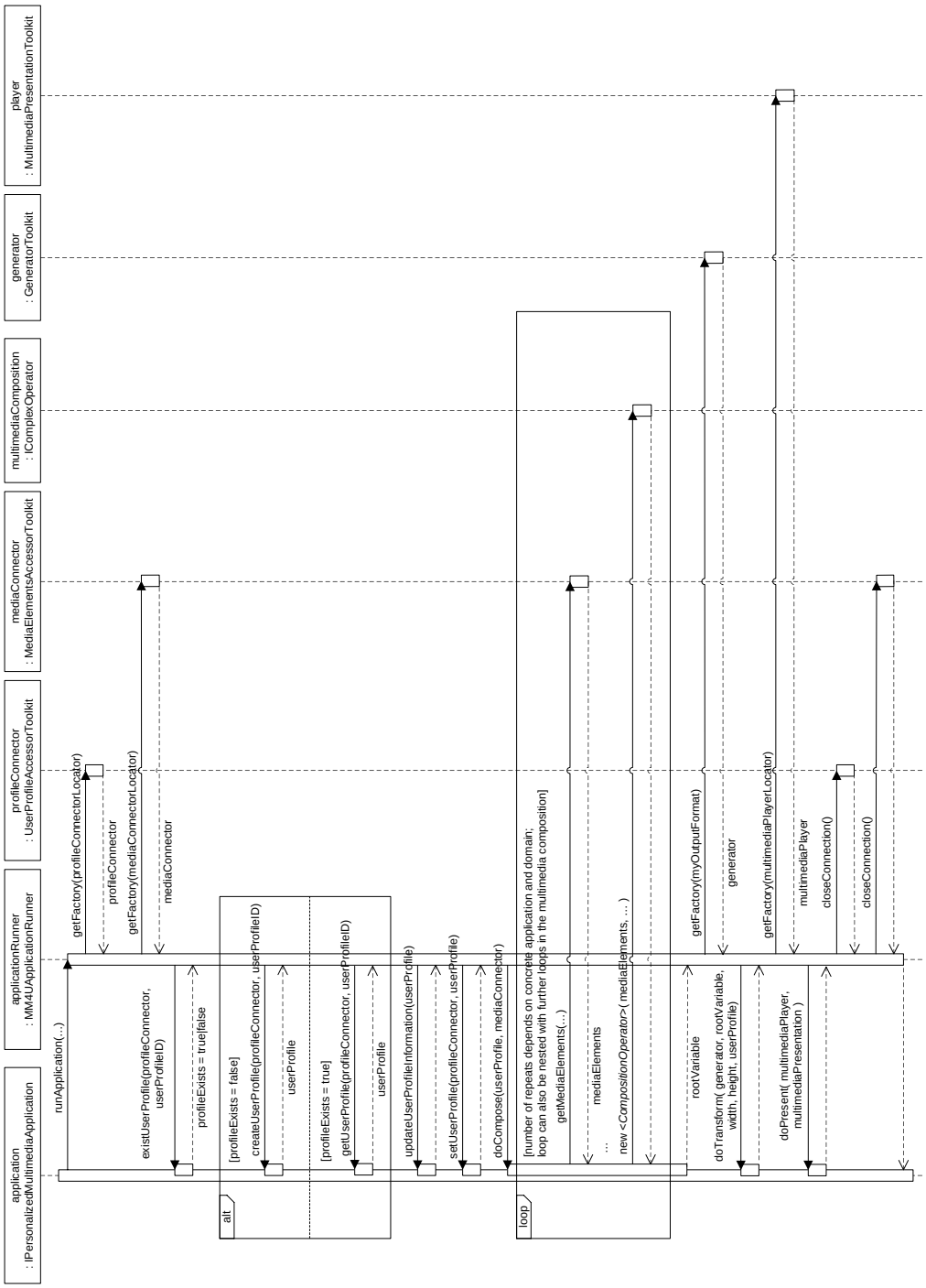


Figure 8.17: Sequence diagram depicting a possible order of how to use the MM4U framework component

To optimize the execution speed of the MM4U framework, the communication between the framework's components is conducted on basis of Java objects. Propagating Java objects between the components ensures for a fast communication of the framework component instances. For example, for implementing the Multimedia Composition and Presentation Format Generators component, the most important design decision in regard of component communication was to implement the MM4U framework's internal multimedia composition model as an object-oriented representation of the personalized multimedia content. This object-oriented representation is used as input model for the transformation into the different concrete presentation formats. By this, we achieve a very fast execution of the multimedia presentation generation process, even on mobile devices.

Another solution for communicating between the Multimedia Composition component and the Presentation Format Generators component could be, e. g., to define a XML-based protocol and thus to exchange XML-documents between the components. However, this would require more memory and computation resources and slow down the generation process of the personalized multimedia presentations.

In addition, for the Presentation Format Generators component, we could have implemented the transformation to the final presentation formats by using, e. g., XSL transformations instead of implementing it straightforward in Java. Here, again the reason for implementing it in Java lies in the performance of such an implementation. Using XSLT would slow down the personalized multimedia content creation process, especially on mobile devices.

8.13.2 Exception Concept

Within any larger software system, it is important to follow a uniform and systematic concept for treating errors [ALRL04]. These errors can lead to a system failure. Java provides with its Exceptions a mechanism for the structured definition and treatment of errors that can occur during runtime. For the MM4U framework, we designed and implemented a simple, however, very flexible and effective exception concept. It bases on an hierarchical organization of MM4U framework specific exceptions. All MM4U framework-specific exceptions start with the prefix MM4U. The central exception interface for the MM4U framework is IMM4UException shown in Listing 8.7. The IMM4UException interface is implemented by the central exception class of the MM4U framework, namely the abstract class MM4UException. The MM4UException class inherits from Java's Exception class. It is responsible for logging all exceptions that are thrown within the MM4U framework and its application-specific extensions. The tracked exceptions are stored in an appropriate log file.

```
public interface IMM4UException {
  public String getObject(); // Object, where an error occurred
  public String getMethod(); // Method causing the error
  public String getComment(); // Additional comments describing the error
}
```

Listing 8.7: The central interface for exceptions within the MM4U framework

Starting with the MM4UException as root class of all MM4U framework related exceptions, a subtyping hierarchy of component specific exceptions is derived.

As depicted in Figure 8.18, for each framework component a specific exception type is determined, e. g., the MM4UMediaElementsConnectorException for the Media Pool Accessor and Connectors component and the MM4UGeneratorException for the Presentation Format Generators component. From these exception classes, further refinements within each component are conducted. These refinements are conducted by further subtyping from the components' central exception class. For example, the MM4UCannotOpenMediaConnectionException is derived from the MM4UMediaElementsConnectorException. It indicates that the connection to the concrete media elements storage solution could not be opened.

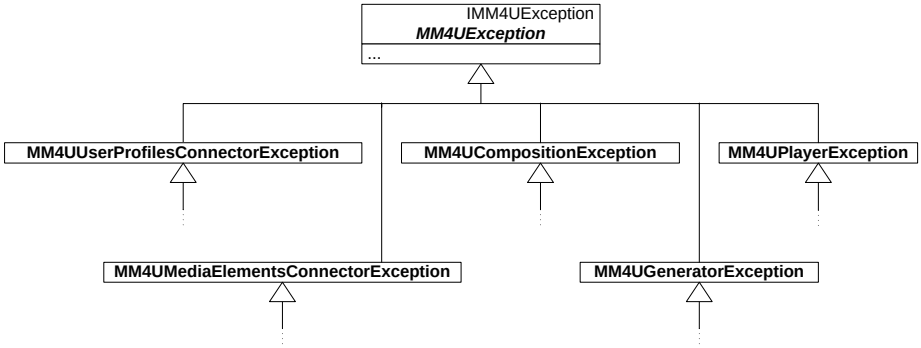


Figure 8.18: Hierarchy of the component specific exceptions of the MM4U framework

8.13.3 Summary

In this section, design issues of the MM4U component framework were considered that are not specific to a single framework component. We considered the aspects of execution performance of the component framework and handling of exceptions. In addition, we also introduced a very slim debugging concept into the MM4U framework. This debugging concept is developed following the Apache Software Foundation's log4j concept [The06b]. However, it does not support sophisticated methods for analyzing the debugging outputs, like it is possible with log4j's Chainsaw.

8.14 Usage of the Multimedia Personalization Framework

The usage of the MM4U framework by a concrete personalized multimedia application is illustrated in Figure 8.19. The personalized multimedia application uses the functionality of the framework to create personalized multimedia content. It integrates whatever application dependent functionality is needed, either by using the already build-in functionality of the framework component instances or by extending the component instances in regard of the specific requirements of a concrete personalized multimedia application (cf. Figure 5.3).

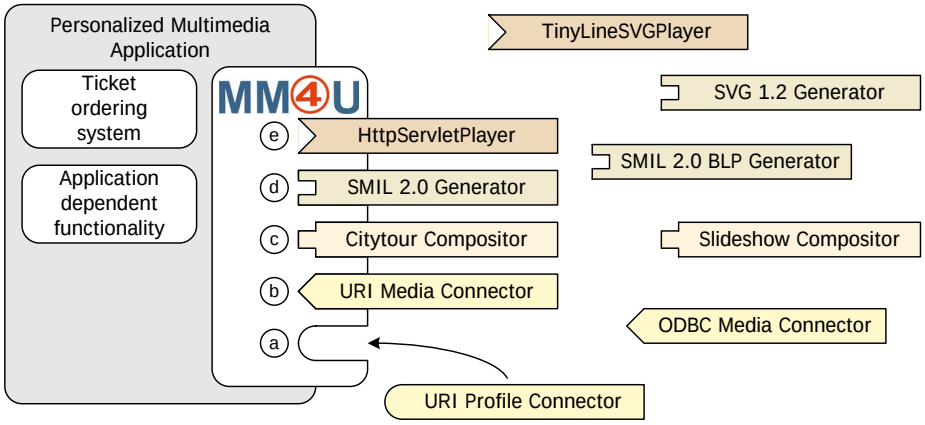


Figure 8.19: Usage of the MM4U framework by a personalized multimedia application

Using the MM4U component framework for developing a concrete personalized multimedia application primarily means composing and reusing the already available instances of the framework's components, i. e., the available component implementations. For example, the existing implementations of the Presentation Format Generators component, the Multimedia Presentation component as well as the components for accessing the user profile information and media elements typically can be reused without modification but by reusing them in different configurations only. However, the Multimedia Composition component is heavily dependent on the actual application's domain. Thus, developers of a personalized multimedia application typically implement their own Multimedia Composition component instance. For it, they reuse the basic multimedia composition functionality the component offers. In addition, any other concrete composition and personalization functionality can be used that is provided by other applications based on the MM4U framework.

As depicted in Figure 8.19, the MM4U framework provides five types of "component hot-spots" where different types of components can be plugged in. Each component hot-spot represents one particular task of the personalization process. The component hot-spots or more formally speaking the variation points of the component framework can be realized by plugging in a component instance that fulfills the contractual specification of the variation point.

To provide application developers a better overview of what is already implemented with the MM4U component framework, the feature diagram shown in Figure 8.20 depicts a feature-oriented view to the MM4U framework's domain of personalizing multimedia content. The figure shows the single features associated to the root-feature of providing for personalized multimedia content. The notation of the feature diagram is *fg*UML taken from [Mei06] and bases—as the name indicates—on UML. The *fg*UML bases on an analysis and improvement of feature graphs as defined with the Feature-Oriented Domain Analysis (FODA) [KCH+90] and its extensions such as [RBSP02] and [GBS01]. Here, a feature is defined as a set of reusable and configurable requirements that helps application developers

with understanding and using the flexible architecture defined by the feature diagram.

The feature diagram for the MM4U framework depicted in Figure 8.20 shows application developers what kind of features exists in the framework and how these features need to be configured for developing a concrete personalized multimedia application. The root-feature PersonalizationOfMultimediaContent is of type concept, which means that it is a conceptual feature of the considered domain. To be more precisely, in this case, the root-feature is equivalent to the considered domain. The root-feature is composed of a set of five extension points and variation points, respectively. Extension points and variation points are specific types of features, where functionality can be adapted and extended for a concrete application. Both extension points and variation points determine the type of the functionality to be adapted or extended. However, the concrete characteristic of the functionality is left open and determined by the concrete applications (see Section 4.1.1).

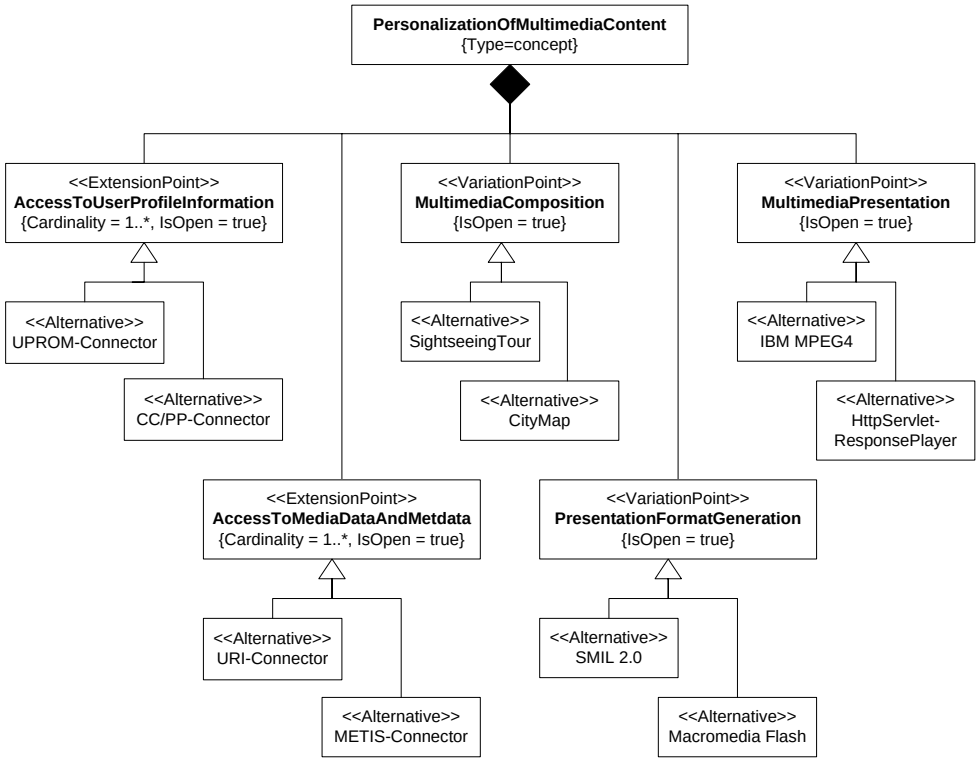


Figure 8.20: Feature diagram of the MM4U framework

As depicted in Figure 8.20, each of the five extension points and variation points corresponds to one of the framework components identified in Section 8.3. The extension points of our MM4U framework are AccessToUserProfileInformation and AccessToMediaDataAndMetadata. They provide access to user profile information and to media data with their associated meta data. As variation points there are Mul-

timediaComposition, PresentationFormationGeneration, and MultimediaPresentation. They represent the multimedia composition functionality of the framework in the internal multimedia content representation model, the transformation of the multimedia content to the concrete output formats, and the playback of the multimedia presentations. Alternatives are the concrete instances of extension points and variation points, respectively. They are added to the feature diagram by using the inheritance notation of UML.

The attribute IsOpen indicates that besides the depicted extension point's or variation point's alternatives, other alternatives can be developed and used by concrete applications. As shown in Figure 8.20, all of the framework's extension points and variation points can be extended by additional functionality. The difference between variation points and extension points is that in the case of variation points *exactly one alternative* must be instantiated by the concrete application at runtime. In the case of extension points, the attribute Cardinality indicates how many alternatives can be simultaneously binded by a concrete application during runtime.

8.15 Deployment of the MM4U Framework Components

By the nature of component-based software, the single components of a component-based system can be deployed in principle anywhere. This serves the requirement to the MM4U framework in Section 5.2.3 to provide for deploying the tasks and phases of the general process chain for authoring personalized multimedia content on different (possibly remotely connected) computers. However, as the deployment of components heavily influences the execution speed of the application, it is reasonable to think about an optimal deployment of the components. For the MM4U framework, in principle two variants of deploying the components are relevant. The first variant is the thin-client depicted by the UML deployment diagram in Figure 8.21. Here, only the Multimedia Presentation component is executed on the user's end device. Thus, we abstract here from the discussion of tightly and loosely coupling the multimedia player software and the Multimedia Presentation component conducted in Section 8.11 for reasons of simplicity. All other four framework components of the MM4U framework reside on a remote server. Thus, a client/server-deployment of the framework components is conducted.

The second variant is the thick-client shown in the UML deployment diagram in Figure 8.22. Here, all five components of the MM4U framework are executed on the (mobile) end device. This enables the concrete personalized multimedia application in principle to work independent from an available network connection. However, it is not limited to offline solutions. If a (mobile) network connection is available, it can also retrieve, e. g., media elements or user profile information via the Internet. The second variant is typically applied for mobile applications with specific requirements in regard of being executable on a mobile end device and usable even if a network connection may be (temporarily) not available. Here, a particular subset of the MM4U framework targeted at mobile devices is employed. This subset is called the mobileMM4U framework and is presented in the following Section 8.16. Both deployment variants of the MM4U framework components have been practi-

cally applied for the development of concrete demonstrator applications presented in Section 10.

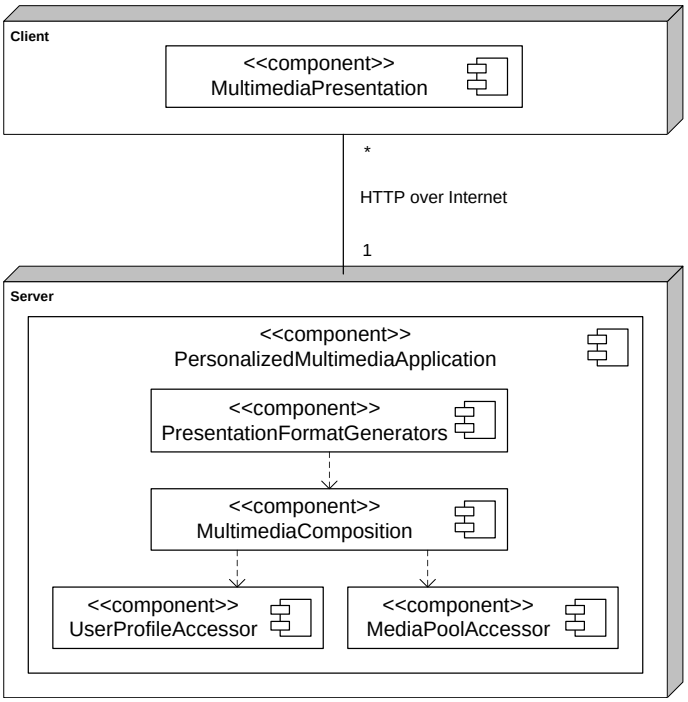


Figure 8.21: UML deployment diagram for the MM4U framework in the thin-client variant

8.16 MobileMM4U—Framework Support for Personalized Mobile Multimedia Applications

With the MM4U component framework presented so far, we have defined a common basis for developing personalized multimedia applications that need to dynamically author and deliver their content in different presentation formats on different (mobile) end devices. However, as stated in the requirements to the MM4U framework in Section 5.2.3, the framework shall not only be executed on a powerful server, providing personalized content for a set of remotely connected (mobile) clients. As personalization is one of the key challenges when dealing with mobile multimedia applications, the framework shall also be executable on mobile devices. Thus, we defined a *mobile subset* of the MM4U framework, which is addressing the very specific requirements that arise in the context of executing the framework on mobile systems. This subset is called the mobileMM4U framework and is implemented in Personal Java and Java 1.1.8, respectively.

The architecture of the mobileMM4U framework is depicted in Figure 8.23. As shown in the figure, the mobileMM4U framework consists of the same layers and

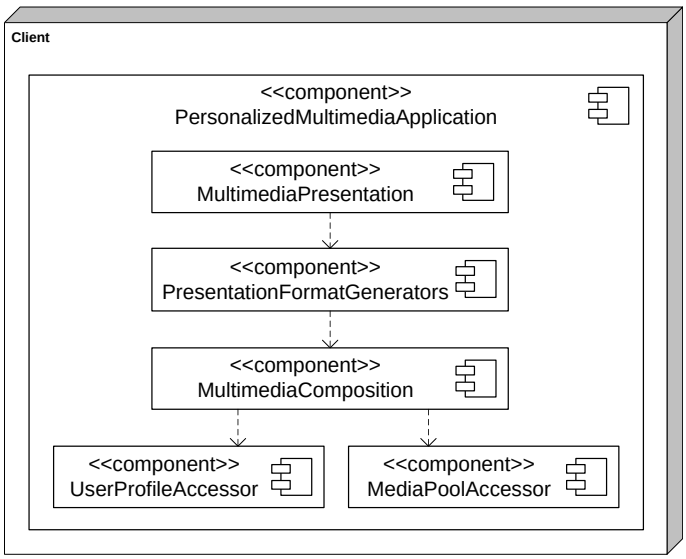


Figure 8.22: UML deployment diagram for the MM4U framework in the thick-client variant

components like the MM4U framework depicted in Figure 8.1. The component instances of the mobileMM4U framework are configured such that they only comprise the specific functionality needed for creating personalized multimedia content for mobile devices [SB04b, BKS04]. This enables the mobileMM4U framework to be installed and executed on such devices. The specific characteristics of the mobileMM4U framework in regard of authoring personalized multimedia content for mobile devices are described in the following along the mobileMM4U framework's components from bottom to top.

User Profile Accessor and Connectors and Media Pool Accessor and Connectors Components The mobile variants of the User Profile Accessor and Connectors and Media Pool Accessor and Connectors component instances provide lightweight connectors to user profile information and media data stored on the mobile device. In addition, the user profile information and media data can also be retrieved from remotely located servers in the Internet via a wireless network connection. For storing and processing user profile information, the hierarchical simple text profile format presented in Section 8.8.2.1 is employed. This is provided by the FilesystemUserProfileConnector for mobile devices. User profile information that are relevant for the mobile users are, e. g., the users' current location and surrounding such as noise level and lighting level or in the case of a mobile tourist application the sightseeing interests and preferences.

For retrieval of media elements, the particularities and characteristics of the mobile device are to be taken into account. Consequently, the mobileMM4U framework provides for a personalized access to media elements suitable for mobile devices. For example, if the display of the device does not support colors, only grayscales media

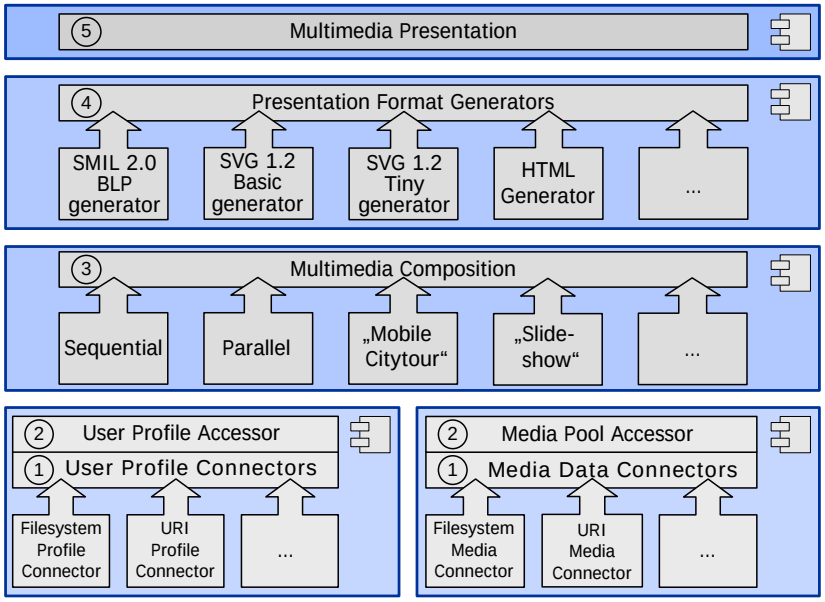


Figure 8.23: Functional and component view of the architecture of the mobileMM4U framework

elements can be selected instead, or if only a particular color depth is supported, the media elements can be adapted to it. If the mobile device is not able to playback a particular media format at all, e. g., a MPEG-compressed video element, it must be appropriately substituted. For example, our personalized mobile tourist guide application presented in Section 10.1 selects by the FilesystemMediaElementsConnector a series of image elements if playback of video elements is not supported.

Multimedia Composition Component In regard of the Multimedia Composition component for mobile systems, the complex and sophisticated composition operators of the mobileMM4U framework have to take contextual information about the users and their mobile device into account. Thus, for our mobileMM4U framework, we developed for example a sophisticated composition operator CityMap for personalized mobile tourist guide applications addressing the requirements of mobile devices. The sophisticated composition operator CityMap supports the presentation of sightseeing spots on a map, which are relevant to the user according to the user profile information. The CityMap operator allows to adapt the size of the map according to the display size of the mobile device. It also automatically adapts the positioning of the selected POIs on the map. Another sophisticated composition operator taking the particularities of mobile devices into account is the SightPresentation operator presented in Section 6.3.5. This operator allows among others to adapt the presentation's layout to the display size of the end device and to select proper media types, e. g., it uses image elements instead of video elements if the end device's hardware or software is not capable of displaying video clips. The CityMap and SightPresentation operators are examples of extending the functionality of the MM4U framework to-

wards mobile application support in the mobileMM4U framework; here in the area of mobile tourism applications.

Presentation Format Generators Component For the MM4U framework, we developed presentation format generators for the different concrete presentation formats such as SMIL and SVG (see Section 8.10.4) targeted at Desktop PCs. With the mobileMM4U framework, we provide a special instance of the Presentation Format Generators component that supports transforming of the internal multimedia content representation model to the specific capabilities of the presentation formats targeted at mobile devices. These mobile multimedia presentation formats consider the limitations of mobile devices by means of building subsets and determining profiles that are particularly designed for usage on mobile devices. For example, we developed presentation format generators for the SMIL 2.0 Basic Language Profile (see [ABC+01b]) that is a particular subset of SMIL 2.0 targeted at mobile devices. The difference between SMIL 2.0 and its mobile profile lies among others in the power of expression of their layouting functionality. While in SMIL 2.0 nested layout regions are allowed, the Basic Language Profile of SMIL 2.0 only admits a flat layout hierarchy (see Section 6.4.4) with an absolute spatial model (see Section 6.3.1.2). This means, that the layout structure of the internal multimedia content representation model, which allows nested regions like in SMIL 2.0, is automatically broken down by the abstract transformation algorithms of the Presentation Format Generators component to a flat one. In addition to SMIL 2.0 BLP, we do also provide support for the mobile profiles of SMIL 2.1 as they are SMIL 2.1 Mobile Profile and SMIL 2.1 Extended Mobile Profile [SMI05]. We also implemented presentation format generators for the two mobile profiles of SVG, namely SVG Tiny and SVG Basic, which are defined in Mobile SVG [AAA+04a]. The SVG Tiny profile is intended for multimedia-ready cell phones and the SVG Basic profile is aimed for pocket computers like PDAs [AAA+04a]. As we cannot assume that there is a mobile multimedia player available on every end user's mobile device, there is also the fall back solution to HTML or XHTML Basic. Thus, if there is no specific multimedia player installed on the mobile device, e. g., a Pocket PC with build in Internet Explorer only, a personalized mobile application can generate pure HTML as output format instead.

Multimedia Presentation Component For the instance of the Multimedia Presentation component of our mobileMM4U framework, we employ multimedia player software designed for mobile devices. For example, we integrated IBM's MPEG-4 player [IBM06a], TinyLine's SVG player [Gir06], as well as a simple HTML viewer [GC96]. However, we also employ external, mobile standalone multimedia players such as Pocket SMIL [INR02] and Pocket SVG [CSI05] for PDAs.

Summary For providers of personalized mobile applications, the mobileMM4U framework supports a more efficient and economic development of such applications. By this, we allow for a shorter and quicker time to market of mobile applications in times in which we find new types of mobile devices every six months. As the component instances of the mobileMM4U framework are seamlessly integrated into the MM4U framework, it is possible to develop applications that dynamically

create personalized multimedia content for both Desktop PCs and mobile devices. An example of an application that serves both Desktop PCs and mobile devices is our personalized (mobile) tourist guide application Sightseeing4U presented in Section 10.1.

8.17 Summary

A special challenge with developing the MM4U framework was to create a general software architecture that takes the multitude of different multimedia personalization aspects into account. Such a generalized solution shall at the same time be practical and easy to use for application developers. The main disadvantage of the also very generalized Standard Reference Model for Intelligent Multimedia Presentation Systems (see Section 3.2.11) is that it is not applicable in practice. The SRM for IMMPS describes the structure of personalized multimedia applications—called intelligent multimedia presentation systems—on the level of different tasks in the multimedia personalization process. For authoring multimedia presentations by these systems, the SRM for IMMPS focuses on knowledge bases and plans. The SRM for IMMPS does not provide software engineering support how to actually design and implement such applications. As a consequence, the application developers need to design and determine the application's architecture by themselves. In contrast, the MM4U component framework provides a generalized architecture for personalized multimedia applications that integrates the different approaches for multimedia personalization as identified in Section 3.3. The MM4U framework defines a standardized application architecture in the domain of personalized multimedia applications. It is practical, directly applicable, and can be easily adapted and extended to the requirements of a concrete application. A framework cookbook provides practical information on how to proceed with the development of a personalized multimedia application on the basis of the MM4U framework (see Appendix A). The presented description and documentation of the MM4U framework architecture provides three important views on the framework architecture (cf. [MMHR04]). These are the structural view provided by class diagrams and component diagrams, the dynamic view with the sequence diagram, and a resource mapping view with the deployment diagrams showing the deployment of the component framework over different machines.

With the MM4U framework and the mobileMM4U framework, we provide generic support for the dynamic generation of personalized (mobile) multimedia presentations. Both, the MM4U framework and the mobileMM4U framework are designed to be independent from any special domain. This means, that they can be used to generate personalized multimedia presentations for any application area like personalized sightseeing guides or personalized e-learning applications. In the following Section 9, the general impact of personalization to the development process of multimedia applications is described. It is shown, how the MM4U framework approach supports this development process. Concrete personalized multimedia applications that have been developed with the MM4U framework and the mobileMM4U framework, respectively, are presented in the section after next.

9 Impact of Personalization to the Development of Multimedia Applications

The multimedia personalization framework MM4U (and mobileMM4U framework, respectively) provides support for developing sophisticated personalized multimedia applications. Involved parties in the development of such applications is typically a heterogeneous team of developers from different fields including media designers, computer scientists, and domain experts. We describe what challenges personalization brings to the development of personalized multimedia applications and how and where the MM4U framework can support the developer team to accomplish their job. The general software engineering issues in regard to personalization are discussed. We describe how personalization affects the single members of the heterogeneous developer team and how the MM4U framework supports the development of personalized multimedia applications.

We observe that software engineering support for personalized multimedia applications such as proper process models and development methodologies are not likely to be found. Furthermore, the existing process models and development methodologies for multimedia applications as, e. g., [RS99] and [ESN03], do not support personalization aspects. However, personalization requirements complicate the software development process even more [FHV02], make development of personalized applications more complex, and thus increase the development costs. Every individual alternative and variant has to be anticipated, considered, and actually implemented. Consequently, there is a high demand in supporting the development process of such applications. In the Section 9.1, we first introduce how personalization affects the software development process in general. In Section 9.2, we identify what support the developers of personalized multimedia applications need and consider in Section 9.3 where the MM4U framework supports the development process.

9.1 Influence of Personalization to the Software Development Process

As the term personalization profoundly depends on the application's context, its meaning has ever to be reconsidered when developing a personalized application for a new domain. Rossi et al. [RSG01] claim that personalization should be considered directly from the beginning when a project is conceived, i. e., already during the requirements and analysis phase. Therefore, the first activity when developing a personalized multimedia application is to determine the personalization requirements, that is, which aspects of personalization should be supported by the actual

application. For example, in the case of an e-learning application the personalization aspects should consider the automatic adaptation to the different learning styles of the students as to be found, e. g., with [BZST04], and the students prior knowledge about a topic (knowledge level) as considered, e. g., in [Par89]. In the case of a personalized mobile tourism application, however, the user's location and his or her surrounding, the user's interests, preferences, and age would be of interest for personalization instead. Other aspects of personalization are described in the application scenarios in Section 2. The personalization aspects must be kept in mind during every activity throughout the whole development process. The decision regarding which personalization aspects are to be supported has to be incorporated in the analysis and design of the personalized application and will hopefully entail a flexible and extensible software design. However, this increases the overall complexity of the application to be developed and automatically leads to a higher development effort including longer development duration and higher costs. Therefore, a good requirement analysis is crucial when developing personalized applications lest one dissipates one's energies in bad software design with respect to the personalization aspects.

9.2 Required Development Support for Personalized Multimedia Applications

When transferring the requirements for developing personalized software to the specific requirements of personalized multimedia applications one can say that it affects all members of the developer team: the domain expert, the media designers, and the computer scientists. Adding personalization to multimedia applications is putting higher requirements to them.

Domain Experts The domain expert normally contributes to the development of multimedia applications by providing input to draw storyboards of the specific application's domain. These storyboards are normally drawn by media designers and are the most important means to communicate the later application's functionality within the developer team. When personalization comes into account, it is difficult to draw such storyboards because of the many possible alternatives and different paths in the application that are implicated with personalization (see our personalized tourist guide example in Section 1.1). Consequently, the storyboards change in regard to, e. g., the individual user profiles and the end devices that are used. When drawing storyboards for a personalized multimedia application, those points in the storyboard have to be identified and visualized where personalization is required and needed. Storyboards have to be drawn for every typical personalization scenario concerning the concrete application. This drawing task should be supported by interactive graphical tools to create "personalized" storyboards and to identify reusable parts and modules of the employed multimedia content.

Media Designer It is the task of the media designer in the development of multimedia applications to plan, acquire, and create media elements. With personalization, media designers have to think additionally about the usage of media elements

for personalization purposes. That means, the media elements have to be created and prepared for different contexts. When acquiring media elements, the media designers must consider for which user context the media elements are created and what aspects of personalization are to be supported, e. g., different styles, colors, and spatial dimensions. Possibly a set of quite similar media assets have to be developed that only differ in certain aspects. For example, an image or video element has to be transformed for different end device resolutions, color depth, and network connections. Since personalization means to (re)assemble existing media elements into a new multimedia presentation, the media designers will also have to identify reusable media elements. This means that additionally the storyboards must already capture the personalization aspects (see above). Not only the content but also the layout of the multimedia application can change depending on the user context. So, the media designers have to create different visual layouts for the same application to serve the needs of different user groups. For example, an e-learning system for children would generate colorful multimedia presentations with many auditory elements and a comic-like virtual assistant, whereas the system would present the same content in a much more factual style for adults. This short discussion shows that personalization already affects the storyboarding and media acquisition. Creating media elements for personalized multimedia applications requires a better and elaborate planning of the multimedia production. Therefore, a good media production strategy is crucial, due to the high costs involved with the media production process. Consequently, the domain experts and the media designers need to be supported by appropriate tools for planning, acquiring, and editing media elements for personalized multimedia applications.

Computer Scientists The computer scientists actually develop the multimedia personalization functionality of the concrete application. What this personalization functionality is, depends heavily on the concrete application domain and is communicated with the domain experts and media designers by using personalized storyboards. With personalization, the design and implementation of the application typically has to be more flexible. This is due to meet the requirements of different and changing user profile information and different end device characteristics. Thus, more abstract classes and interfaces need to be specified to provide the different personalization aspects of the application. This more flexible and abstract design makes it difficult in the design phase of the application to anticipate the application's runtime characteristics. The requirement for providing a flexible software architecture and the challenges involved with it come along with the challenges of developing runtime-adaptive systems in general. Such systems need to provide interfaces to be able to dynamically adapt their behavior to changing requirements during runtime.

9.3 MM4U Framework Support for Developing Personalized Multimedia Applications

In respect of the software development process of personalized multimedia applications, the MM4U framework assists the computer scientists during the design and

implementation phase of their personalized multimedia application. It alleviates the time-consuming multimedia content assembly task and lets the computer scientists concentrate on the development of the actual application-specific functionality. The MM4U framework provides functionality for the single tasks of the personalization process as described in Section 5.1. It offers the computer scientists support for integrating and accessing user profile information and media data, selecting media elements according to the user's profile information, composing these elements into coherent multimedia content, and generating this content in standardized multimedia presentation formats to be rendered and displayed on the user's end device.

The MM4U framework provides the computer scientists the general, domain independent architecture of the personalized multimedia application and supports them in designing and implementing the concrete multimedia personalization functionality. The MM4U framework relieves application developers from the time-consuming tasks in the context of multimedia content composition and personalization in order to let them concentrate on the development of the application-specific functionality. Consequently, application developers require extensive support for creating personalized multimedia content. For it, the MM4U framework provides substantial support for the dynamic generation of arbitrary personalized multimedia content in respect of, e. g., the user's interests and preferences, the current location and environment, as well as the used end device. When using the MM4U framework, the computer scientists must know how to use the existing component instances and how to extend the framework functionality.

As described in Section 8.14, using the MM4U component framework mainly means composing and reusing the already available instances of the framework's components. Typically, the instances of the Presentation Format Generators component, the Multimedia Presentation component as well as the components for accessing the user profile information and media elements can be reused without modifying the source code of the component instances. Besides reusing these existing component instances, the development of personalized multimedia applications by using the MM4U framework means to the computer scientists to develop new complex and sophisticated multimedia composition operators for generating personalized content. Here, the framework provides the basis for developing such complex and application-specific multimedia personalization functionality, as for example the Slideshow and the SightPresentation operators presented in Sections 6.3.4 and 6.3.5. The domain experts provide the knowledge about the application domain and therewith the input for the computer scientists for developing the application-specific multimedia composition functionality. The media designers' role in developing an application with the MM4U framework is to create the media objects (possibly in different variants required for personalization) and develop the multimedia layout.

To assist the computer scientists methodically in applying the MM4U component framework for the development of their personalized multimedia application, a framework cookbook is provided that describes in form of guidelines and checklists the tasks that have to be conducted to apply the framework. An excerpt of this cookbook is presented in Appendix A.

Every concrete personalized multimedia application itself provides by its instance of the Multimedia Composition component new multimedia composition and personalization functionality to the MM4U framework. By this, reuse of existing

personalization functionality is increased. Consequently, the MM4U framework improves the development process of personalized multimedia applications and supports for a more efficient and economic development of such applications (see process improvement requirements in Section 5.2.3). Furthermore, the design of the Multimedia Composition component enables the framework to embrace existing approaches for generating personalized multimedia content as presented in Section 3.3. For it, the approaches only need to provide support for generating multimedia document trees in the internal multimedia content representation model that can be created with the basic and complex composition functionality of the MM4U framework.

10 Usage of the MM4U Framework in Real Applications

The usage of the MM4U framework for the development of demonstrator applications is the second evaluation step of the ProMoCF approach. In the first step, the ProMoCF approach has been applied for the development of the MM4U component framework. Now, the outcome of this employment, i. e., the MM4U framework itself is applied and evaluated by employing it for the development of real personalized multimedia applications. The development of these applications gave us important feedback about the comprehensiveness and the applicability of the framework. At the same time, it provided feedback for improving the process model and development method ProMoCF for component frameworks.

For example, during the development of the MM4U framework and the first concrete demonstrator applications employing this framework, we observed that the granularity of hot-spot-cards (and the granularity of hot-spots, respectively) of about one method as defined in Pree's original hot-spot-driven design process is too imprecise. Comparing the flexibility requirements written on hot-spot-cards were difficult in order to identify groups of cards which have a high affinity in regard of solving a particular problem in the framework's domain. Due to the imprecise definition of hot-spot-cards, it was not clear how much and which functionality actually corresponds it. Thus, it was difficult to design and to decide how to employ the MM4U component framework for the development of concrete demonstrator applications, as this appliance requires clearly specified and unambiguous interfaces. This leads to the definition of the granularity of hot-spots to be exactly one (hook) method in the programming language (see Section 7.2).

In addition, the initial version of the ProMoCF approach did not already come with the concept of group-hot-spot-cards. Once, we defined the granularity of hot-spot-cards to be exactly one hook method, we could clearly compare the flexibility requirements to the framework written on these cards. This allows us to identify groups of hot-spot-cards that describe flexibility requirements for a common problem in the framework's domain. With evolving the MM4U framework and developing further applications as well as improving existing applications employing the framework, it emerged that the identified groups of hot-spot-cards are likely corresponding to the components of the MM4U framework. This means, that each identified group of hot-spot-cards is encapsulated in the MM4U framework by a distinct framework component. To make this knowledge explicit, we introduced the concept of group-hot-spots and group-hot-spot-cards, respectively. Thus, a group-hot-spot-card is targeted at identifying the distinct components within a component

framework and defines the flexibility requirements to these components (see Section 7.2).

With the development of concrete personalized multimedia applications employing the MM4U component framework in different domains, the applicability and with it the quality of the framework has been proved. These personalized multimedia applications are described in the following sections. In Section 10.1, the generic personalized mobile city guide application Sightseeing4U is described. In the subsequent Section 10.2, the personalized multimedia sports news ticker Sports4U is presented. In Section 10.3, Manual4U is introduced, an application providing interactive instruction manuals on mobile devices. A multi-channel multimedia content creation service Transformation4U is presented in Section 10.4. In Section 10.5, the Pictures4U application is described, providing personalized photo albums in the Internet. So far, all applications are targeted at an automatic authoring of the personalized multimedia content. In Section 10.6, we present a context-driven smart authoring tool *x*SMART that allows for the semi-automatic creation of personalized multimedia content by domain experts. It can be extended by arbitrary application-specific and domain-specific authoring wizards that guide domain experts through the context-driven multimedia authoring process. As an example of such an authoring wizard, the Pictures4U wizard is presented in Section 10.6. This wizard is derived from the Pictures4U application presented in Section 10.5. However, it provides for a semi-automatic authoring of personalized photo albums.

For each application presented in the following, we briefly present the considered application area. We describe the usage of the MM4U component framework by the personalized multimedia application. This means, we present which framework components are actually used by the application, how is the framework adapted to the specific requirements of the application-domain, and what other particularities we were faced with developing the application. In addition, the impact of the application for the specific application-domain is considered.

10.1 Sightseeing4U—A Generic Personalized City Guide Application

- Name: Sightseeing4U
- Application area: travel and tourism
- Usage of the MM4U Framework: uses the entire MM4U framework and mobileMM4U framework
- Impact: efficient development of mobile sightseeing guides

Mobile applications such as mobile tourist guides that provide tourists with location-based information today mostly aim at adapting the multimedia content to the different end user devices. More and more, these applications also exploit positioning information like the Global Positioning System (GPS) to guide the user on the trip. What is still lacking, however, is the availability of *personalized* and also *multimedia-rich* (mobile) content (cf. Section 3.2.9). In particular, user profile information shall be exploited to provide for a personalized selection, creation, and delivery of multimedia content that meets the individual user's needs, preferences, interests, and (mobile) device characteristics. For mobile applications, the user's

profile information plays an important role which can be exploited to best meet the user's individual information needs at the mobile client.

However, as described in Section 9.1 such a personalization support increases the application's complexity since every individual alternative has to be considered and implemented. Consequently, to provide substantial support for the development of personalized (mobile) multimedia applications, we developed a generic mobile sightseeing guide on the basis of the MM4U framework and mobileMM4U framework, respectively. This generic tourist guide is called Sightseeing4U [BKS04, SB04a, BBK+04] and is aimed at providing generic support for developing personalized (mobile) multimedia tourist applications for arbitrary cities and landscapes.

On basis of the generic Sightseeing4U application, we developed a client/server-architecture of a personalized multimedia sightseeing information system. This architecture is presented in the following Section 10.1.1. In addition, we developed two concrete instances of our generic tourist guide application Sightseeing4U for our hometown Oldenburg in Northern Germany. These two instances are a thin-client variant and a thick-client variant, following the deployment of the MM4U framework components described in Section 8.15. Both, the thin-client and the thick-client variant of our Sightseeing4U instance for Oldenburg are integrated into our architecture for personalized multimedia sightseeing information systems presented in Section 10.1.1. In Section 10.1.2, the features of the generic Sightseeing4U application for personalized mobile tourist guides are presented, before we come to the description of the two client variants in Section 10.1.3 and 10.1.4. With the client/server-architecture and the generic Sightseeing4U application, we aim at addressing the limitations of mobile devices, the unreliability of the network, and the different user's interests for the selection, composition, and delivery of personalized multimedia content to mobile and stationary devices.

10.1.1 Architecture for a Personalized Multimedia Sightseeing Information System

In this section, we draw our attention to some general architectural issues to show how personalized multimedia content can be generated and deployed to different (mobile) devices. In the domain of mobile computing, location-based multimedia content can be retrieved either from media storages on the mobile device itself or from remote servers in the Internet. Here, the content reflects the user's current position. Providing for personalized multimedia content adds an additional challenge, as the personalized application needs access to the potentially relevant media elements for the individual users. The limited resources of a mobile device and the potentially unavailability of a wireless network connection influence the generation and distribution of the personalized multimedia content.

In order to provide such an individualized information delivery, the system architecture presented in Figure 10.1 has been developed. In an optional pre-trip phase the Desktop PC is used to prepare the trip. Depending on the capabilities of the device and the network conditions, the architecture allows to pre-generate personalized content for the end user's trip (thin-client variant) or leave this open to the trip (thick-client variant). Following the trip, a post tour analysis can be carried out. Technically, the presented architecture addresses the heterogeneity of mobile

devices with regard to computing power, memory, network bandwidth, and changing network availability. It is based on a sightseeing server, which gives access to multimedia data and profile information.

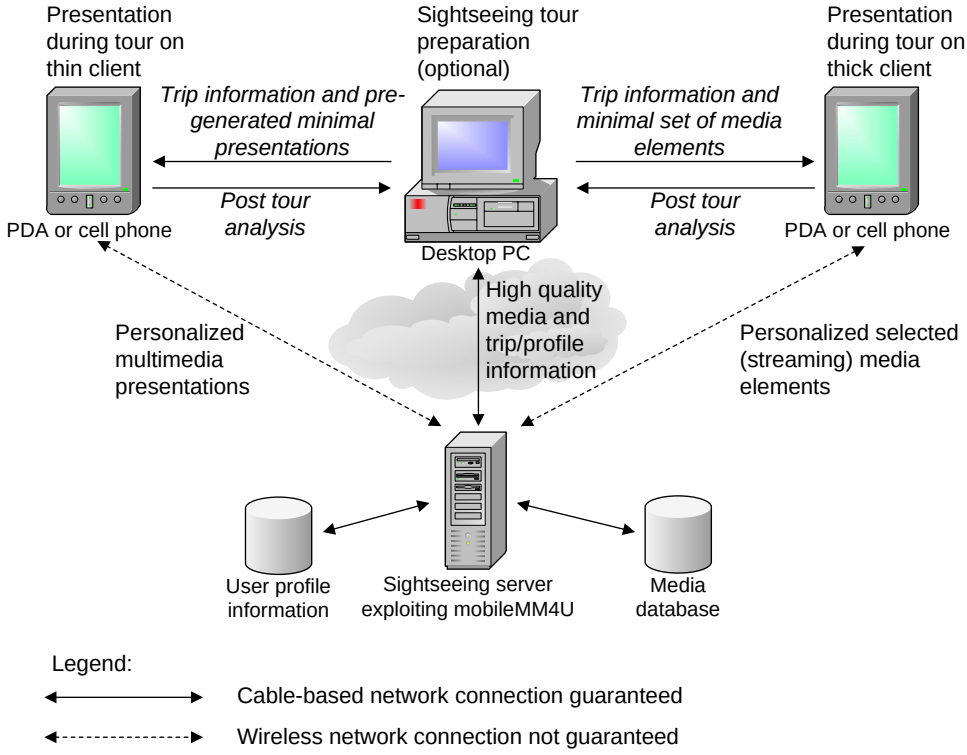


Figure 10.1: The client/server-architecture of a personalized multimedia sightseeing information system based on Sightseeing4U

The architecture in Figure 10.1 assumes a permanent network connection between the client and the server, which is not a disadvantage in general. But for mobile applications and services this may not be appropriate because of potentially high network charges and possibly small network bandwidth. Also network failures in areas with no reception must be taken into account by the client application. For example, the personalized city tour should not just stop processing and returning a general network error message to the user when the connection is lost. To be more independent of the network connection it is inevitable that at least some parts of the application logic reside on the client device. That is, the application logic is shifted from the server to the client and media data is buffered in client's caches. The amount of available memory and processing power of today's mobile devices is already sufficient enough for it. With the application logic residing on the client side, the user profile information, the sight's media data with associated meta data, as well as further information such as the sight's location and categories are

not read from the Internet using a wireless network connection. However, they are stored in and read from locally cached databases and repositories.

For the connectivity of the mobile client, we assume a wireless network connection as default channel to receive multimedia presentations and/or single media elements. The availability of a wireless network connection allows the client to retrieve personalized multimedia presentations or individually selected media elements from the sightseeing server that reflect the latest user profile information. If the network connection is lost, the tourist guide can use the pre-generated presentations stored on the mobile end device as fall back. Figure 10.2 summarizes how the client architecture (thin or thick) and the availability of a wireless network connection influence the generation and distribution of the multimedia content.

	Wireless network connection available	**No wireless network connection available**
Thin client architecture	Multimedia presentations are retrieved from Sightseeing4U server	Pre-generated, pre-stored minimal multimedia presentations
Thick client architecture	Media elements are retrieved from network and presentation generation on client-side	Presentation generation on client-side using a minimal set of pre-stored media elements

Figure 10.2: Impact of client architectures and network availability to the content generation

10.1.2 Features of the Generic Application for Personalized Mobile Tourist Guides

In the previous section, we presented the architecture of a sightseeing information system based on Sightseeing4U. In this section, the features of the generic tourist guide application are presented. The generic tourist guide application Sightseeing4U is applicable for the development of personalized tourist guides for arbitrary cities and landscapes. It is designed to be executed on both Desktop PCs and mobile devices. With the generic Sightseeing4U application, personalization of the sightseeing spots, i. e., the city's or landscape's points of interests (POIs), is supported in regard of arbitrary definable user interests and preferences. Depending on the specific sightseeing interests of an individual user, the generic Sightseeing4U application automatically selects the proper sights for the user. This is realized by category matching of the user's interests with the meta data associated to the sights. The selected POIs the user is—according to the user profile information—interested in are presented as sightseeing spots on a map of the concrete tourist guide's city or landscape. When clicking on a certain spot, the user receives a multimedia presentation with further information about the sight. With regard to the location, the users can also choose whether to automatically receive a multimedia presentation of the sight they are currently approaching.

For authoring a multimedia presentation for a selected sight, automatically those media elements are chosen that best fit the (mobile) end device's characteristics. For example, a user sitting at a Desktop PC receives a high-quality video about a selected sight from the Media Pool Accessor and Connectors component, while a

mobile user gets a smaller video of less quality. If there is no video of a particular sight available at all or if the used end device is not capable of rendering video elements, the generic personalized tourist guide automatically selects relevant images instead and generates a slideshow for the user. In regard of mobile devices, their limitations also effect the layout of the mobile sightseeing presentations. Consequently, not only media elements of lower resolution and size are selected, but also the layout of the multimedia sightseeing presentations is changed, e. g., the heading line of the sight's presentations used for Desktop PCs is removed for the mobile devices.

Since not all multimedia players are available on all end devices and not all presentation formats are suitable for all of these devices, the generic tourist guide employs the (mobile) MM4U framework's functionality to provide presentations in different output formats such as SMIL, SVG, MPEG-4, and Flash, as well as their corresponding mobile profiles. This enables the generic tourist guide application to deliver personalized multimedia content for different types of (mobile) end devices. If there is no proper multimedia player installed on the user's end device at all, there is the fallback to HTML, e. g., by employing the build-in Internet Explorer available on Pocket PCs.

The generic tourist guide application Sightseeing4U also provides for changing the presentations' language according to the user's preference. For it, arbitrary languages can be defined within the generic application. As far as appropriate media elements are available for the selected language, i. e., textual descriptions or spoken audio files of the sights exists, these are selected and presented to the user. If there are no media elements available for the user's preferred language, the generic Sightseeing4U application switches to a predefinable default language instead. Besides changing the language, the generic Sightseeing4U application also allows for using different modalities for the sight presentations. Thus, instead of textual descriptions of the POIs also spoken audio files can be employed. Which modalities are used for the sight presentations is determined by the user's preferences. In addition, the users can select their preferred presentation style. Here, the generic Sightseeing4U application allows for defining arbitrary styles a concrete instance of the tourist guide can provide, e. g., classic, fancy, or children style.

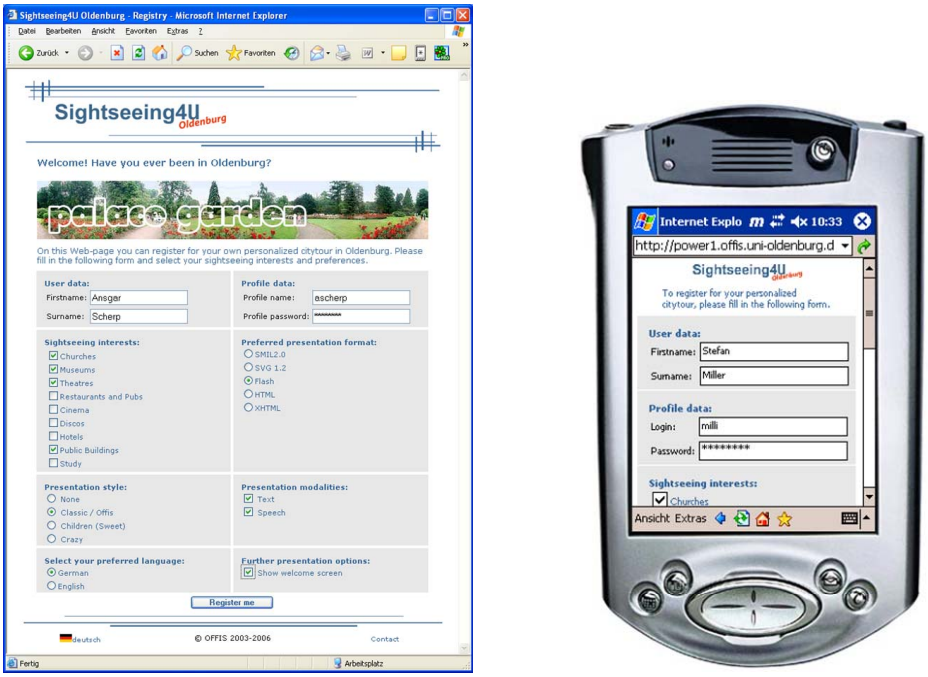
For our hometown Oldenburg, we developed both a thin-client variant and a thick-client variant of the generic tourist guide application These two concrete instances mainly consider the city center. Both variants comprise more than 50 POIs and provide for each POI a textual description as well as image elements in form of photos. The thin-client variant also provides support for video clips. In addition, both client variants of our Sightseeing4U application for Oldenburg support personalization in respect of the user's interests such as churches, museums, theaters, restaurants, cinemas, hotels, and public buildings as well as user's preferences in regard of the favorite language. However, only the thin-client variant also provides support for the preferred modality and presentation style. In the following Section 10.1.3, we first present the thin-client variant of our Sightseeing4U instance for Oldenburg. In Section 10.1.4, we describe the thick-client variant.

10.1.3 Thin-client Variant of Sightseeing4U for Oldenburg

The thin-client variant of our Sightseeing4U instance for Oldenburg employs the MM4U framework located on the server of our architecture for a personalized multimedia tourist information system (see Section 10.1.1). As described in the architecture section, for the thin-client variant the personalized multimedia presentations are not created on the device itself but generated on-demand by the remote sightseeing server. These generated presentations are then transferred via a wireless network connection to the client device. The Sightseeing4U application logic employing the MM4U framework is installed and executed as Java servlet on a Unix server connected to the Internet. The servlet is provided by the Apache Tomcat web server [The06a]. The communication between the clients and the server is conducted via HTTP and by employing the HttpServletPlayer of our Multimedia Presentation component instance presented in Section 8.11.2. For using the thin-client variant, the users need to register by filling out the form depicted in Figure 10.3. Here, the users choose a login and password and state their individual sightseeing interests and preferences. Having successfully registered, the users first receive a map of the city of Oldenburg together with the sightseeing spots on it that meet their sightseeing interests. This personalized city map is authored by the sophisticated composition operator CityMap presented in Section 8.16. Starting from this map, the users can interactively explore the sights of Oldenburg. This means that the users can select a sight on the map and the client sends a request to the Sightseeing4U server. Here, a multimedia presentation about the sight is generated according to the user's preferences on-demand as described in Section 10.1.1. Then the presentation is sent back to the client and displayed to the user. For storing the user profile information on the Sightseeing4U server, the XMLUserProfileConnector of our User Profile Accessor and Connectors instance is employed (see Section 8.8.2.2).

The client device can be either a mobile device, such as a PDA or cell phone or a stationary Desktop PC. The Desktop PC version is used for pre-trip planning and post-trip analysis, whereas the mobile device is actually guiding the tourist during the trip. For the case that there is a wireless network connection failure during the trip, all presentations can also be pre-generated and copied to the mobile device prior to the actual tour. These presentations may nevertheless not reflect the current situation due to their prefabricated nature. In both cases, the computing power for the presentation generation is only needed at the server. The client just renders the presentations.

Figure 10.4 depicts some screenshots of our thin-client variant of the Sightseeing4U instance for Oldenburg in different output formats and on different end devices. The different interests of the users result in different spots presented on the map of Oldenburg's city center. When clicking on a certain spot, the users receive a multimedia presentation with further information about the corresponding sight (see the smaller screenshots where the arrows point at). The presentation in Figure 10.4(a) employs the crazy style, is targeted at a user interested in culture, and comprises a description in German. The presentation in Figure 10.4(b) uses no additional style, is generated for a mobile user who is hungry and searches for a good restaurant in Oldenburg. Here, the presentation is supported by an auditive narration in English. Whereas Figure 10.4(a) shows presentations in SMIL 2.0



(a) (b)

Figure 10.3: Register screen of the thin-client variant of Sightseeing4U on both (a) Desktop PC and (b) PDA

format employing the RealPlayer [Rea06] on a Desktop PC, Figure 10.4(b) depicts presentations in the SMIL 2.0 BLP format using the PocketSMIL Player [INR02].

In addition, a user sitting in front of a Desktop PC receives a high-quality video about the palace of Oldenburg as depicted in Figure 10.4(a), while a mobile user gets a smaller video of less quality in Figure 10.4(b). In the same way, the user searching for a good restaurant in Oldenburg receives a high-quality video when using a Tablet PC as depicted in Figure 10.5(a). However, as video elements are not supported by the mobile player in Figure 10.5(b), a set of image elements are selected instead. The screenshots depicted in Figure 10.5(a) show presentations in Flash format employing Adobe's Flash plug-in [Ado06c]. The presentations shown in Figure 10.5(b) are in Mobile SVG format and are taken from the Pocket eSVG viewer [Exo05] on a PDA. While the classic style is used for the presentation in Figure 10.5(a), no additional style is employed for the screenshot in Figure 10.5(b).

10.1.4 Thick-client Variant of Sightseeing4U for Oldenburg

With the thick-client variant of our Sightseeing4U instance for Oldenburg, the multimedia presentations are generated on the client device according to the user's profile information and by retrieving the media elements from the wireless network.

Figure 10.4: Screenshots of the Sightseeing4U city guide application for a user interested in culture

If there is no network connection, the minimal media assets copied to the mobile device beforehand are used to create the actual presentations on the mobile device applying the mobileMM4U framework. Here, the FilesystemMediaElementsConnector of the Media Pool Accessor and Connectors instance of the mobileMM4U framework is employed. When a network connection exists, more sophisticated and up-to-date media elements reflecting the user's profile information are retrieved from the remote sightseeing server. To overcome the download time, the presentation nevertheless starts with an "introduction" exploiting the minimal media assets.

The Thick-client Variant's Foundation The foundation of the thick-client variant is very different to the thin-client variant presented in the previous section. Instead of employing the MM4U framework on a server machine, the mobileMM4U framework is used on the client device. In addition, the thick-client variant of the Sightseeing4U instance of Oldenburg is integrated in the Niccimon platform for mobile location-aware applications. The Niccimon platform is briefly described in the following. Within the Niccimon project[1], a modular platform is developed that supports the rapid development and deployment of location-aware mobile applications [BBK+04, KBB04, RGB+03, OFF06]. The architecture of the Niccimon platform is depicted in Figure 10.6. The top of the figure shows the core of the Niccimon platform consisting of a mediator and communication interface. Here, arbitrary modules for mobile applications can be plugged into the Niccimon platform. The communication and cooperation between the different modules and the modules'

[1] Competence Center of Lower Saxony for Information Systems for Mobile Usage (Niccimon) funded by the Ministry for Science and Culture of Lower Saxony in March 2001.

(a) (b)

Figure 10.5: Screenshots of the Sightseeing4U city guide application for a user searching for a good restaurant

life cycle is controlled by the platform's mediator. The existing modules of the Niccimon platform provide typical functionality for the specific requirements of mobile applications, such as support for location-aware mobile navigation and orientation, multimodal user interfaces, management of POIs, and location-based information and services. Diverse modules of the Niccimon platform are exploited for the thick-client variant of our generic tourist guide for Oldenburg. These modules provide for determining the location of the user via GPS (Location module) and presenting the user's location and the POIs on a map (GIS module and POI module). For dynamically creating personalized multimedia presentations, we developed another module that integrates the mobileMM4U framework and its application-specific extensions towards the generic Sightseeing4U application (mobileMM4U module).

Implementation of the Thick-client On the basis of the Niccimon platform and the newly developed mobileMM4U module, we developed the thick-client variant of our Sightseeing4U application for Oldenburg. Employing the mobileMM4U framework, the thick-client variant provides for dynamically creating multimedia presentations in different mobile presentation formats like SMIL 2.0 BLP, Mobile SVG including SVG Basic and SVG Tiny, and HTML. It is implemented using Personal Java and by this it is executable on Java-enabled PDAs and cell phones. The thick-client variant employs the TinyLine SVG player [Gir06] for presenting the personalized multimedia content. In addition, it can also employ other multimedia players the Multimedia Presentation instance of the mobileMM4U framework provides such as IBM's MPEG-4 player [IBM06a] and a simple HTML viewer [GC96]. We implemented the thick-client variant of the generic tourist guide application on a Pocket PC with Windows Mobile 2003 and Jeode VM.

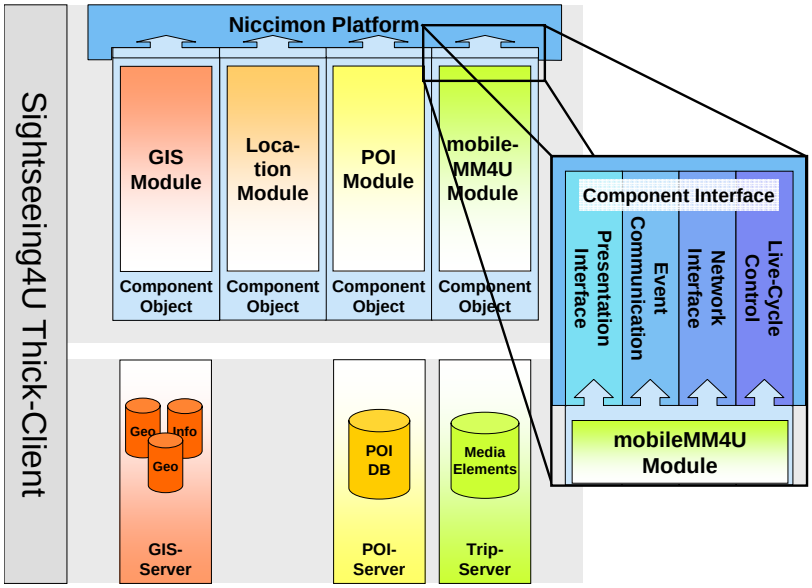


Figure 10.6: The modular Niccimon platform for location-aware mobile applications

Figure 10.7 shows some sample screenshots of our thick-client variant of the Sightseeing4U instance for Oldenburg. With the thick-client variant, the users do not have to register and log in for using the Sightseeing4U application. Thus, when starting the city guide application the map of Oldenburg is shown as depicted in Figure 10.7(a). If the end device is connected with a GPS receiver, also the user's position is displayed on the map. This is shown in the figure by the flag encircled with the compass rose. The users can select their sightseeing interests and preferences as shown in Figure 10.7(b). These are stored within the thick-client variant by employing the simple key-valued-based text format for user profile information provided by the FilesystemUserProfileConnector (see Section 8.16). Then, like with the thin-client variant, the POIs are shown on the map according to the users' interests. This is depicted in Figure 10.7(c). However, here the POI module of the Niccimon platform is employed rather than the sophisticated composition operator CityMap. When the users click on one of the presented POIs, a multimedia presentation about the selected sight is generated according to the users' preferences. Two examples of such multimedia presentations are depicted in Figures 10.7(d) and (e). Besides activating the presentations by hand, they are also activated when the users are changing their physical location and getting close enough to a POI such that a certain proximity threshold is reached. Here, the "position"-context of the user activates the proximity triggered POIs. Thus, these POIs are called context-aware POIs (*x*POIs) [KB05]. With the presented thick-client variant of the generic tourist guide application, we integrate the support for creating personalized multimedia presentations with mobile location-based services.

Figure 10.7: Screenshots of the thick-client variant of the Sightseeing4U instance for Oldenburg

10.1.5 Summary

The presented sightseeing information system architecture addresses the specific demands of personalized multimedia tourist guide applications. Depending on the availability of a wireless network connection, a user can go on the trip with pre-generated personalized presentations but also generate those just on-demand while being on the way. Embracing the tourist life cycle, a user can use pre-planning to pack the potentially relevant media information to the mobile device. Additionally, a post processing of the tour can take place, e. g., the trip's path can be logged and annotated with pictures taken by the user's digital camera.

Based on our MM4U framework and mobileMM4U framework, respectively, we developed with Sightseeing4U a generic tourist application that is applicable to any city and landscape where people wander around and follow tours. With the generic Sightseeing4U application, we provide for an efficient and cost-effective de-

velopment of personalized mobile multimedia tourist applications. We developed an instance of the Sightseeing4U application for our hometown Oldenburg in two variants, the thin-client and the thick-client. While the thin-client employs the MM4U framework executed on a server, the thick-client applies an integration of the mobileMM4U framework and the Niccimon platform for location-aware mobile application.

10.2 Sports4U—A Personalized Multimedia Sports News Ticker

- Name: Sports4U
- Application area: sports news
- Usage of the MM4U Framework: bayesian networks for managing user profile information, MediÆther for media data
- Impact: automatic classification, selection, and composition of sports news

Searching for relevant information in the web today is more and more time consuming. Applications that support their users in searching and retrieving information such as sports news which are relevant and interesting to the users can alleviate this task. Thus, the second demonstrator application employing the MM4U framework is a personalized multimedia sports news ticker called Sports4U. The Sports4U application provides users interested in sports a dynamically generated personalized multimedia sports news ticker. The aim of the Sports4U application is to select and present those sports news out of a pool of sports events that are most relevant and interesting to the users.

Technical Foundation and Implementation Besides the MM4U framework, the Sports4U application exploits the peer-to-peer multimedia event space MediÆther [BW03]. This event space is connected to the MM4U framework via the MediAEther-MediaElementsConnector introduced in Section 8.7.3.2. For the Sports4U news ticker application, the MediÆther event space forms the basis for storing and managing the sports events. A multimedia sports event stored in the MediÆther comprises information about the event such as the event's time and location, the people involved, and to which kind and category of sports the event is associated. With kinds of sports it is meant sports such as soccer, tennis, swimming, rowing, or dancing and categories of sports are, e. g., winter sports, motor sport, or martial arts (cf. sports genres in [Wit99]). In addition, the event type is provided such as Olympic Games, World Football Championship, Wimbledon Championship, and others. A multimedia sports event in the MediÆther also comprises as meta data the different media elements associated to the event such as a title, a textual description, one or more photos, an audio record, or a video clip. However, the actual media data are not stored within the MediÆther but in external stores.

The media data connector of the MediÆther notifies the Sports4U application when new sports events emerge in the MediÆther. Thus, a server-push retrieval of new sports events and the delivery of the corresponding media elements is used as communication direction between the Sports4U application and the MediÆther

event space. Depending on the user profile information, the MediÆther preselects those sports events from the pool of events that best match the user's interests.

The personalized sports news ticker application combines the multimedia data of the selected sports events, employs the available meta data as well as additional information, e. g., from a soccer player database, to generate the multimedia sports news ticker presentation. The multimedia sports news ticker presentation consists of a set of smaller sports presentations, each showing a distinct sports event. These single sports event presentations are arranged in a sequence. When the playback of the ticker reaches the end of the last sports event presentation in the set, the playback of the ticker is restarted from the beginning. For authoring the sports news ticker presentation, the Sports4U application employs a sophisticated composition operator to create the sequence of the single sports event presentations. Here, the Sports4U application provides for taking different constraints into account such as time restrictions for the duration of the news ticker in total as well as for the single events. The single sports events are selected according to the user profile information. For creating the presentations of the single sports events, different presentation templates are defined. These presentation templates are provided by different sophisticated composition operators that allow for automatically arranging the media data of the sports events into a coherent sports event presentation. The provided templates employ different media types for assembling the sports event media content. Thus, different types of templates can be chosen such as a template that presents the sports event by employing an image element and text element only, a template that uses a video element with headline, or a template that uses a series of image elements and a corresponding audio element. Which template is used for presenting a specific sports event is determined according to the user's preferences as well as the media elements available for the specific sports event. For example, if the user has a low bandwidth connection and thus does not want to receive video elements, a series of images is shown instead. If a sports event is delivered with a textual description and a single image element only, a template requiring a video element cannot be applied but a template employing the provided media types of text and image.

The sports interests of the users are modeled within the Sports4U application by employing bayesian networks [Jen01]. The concrete user profile describes the values that are used by the bayesian network to actually determine those sport events provided by the MediAEtherMediaElementsConnector that are added and presented in the personalized multimedia sports news ticker. To manage these user profile information, we integrated an XMLBayesUserProfileConnector into our instance of the User Profile Accessor and Connectors component. This connector stores the user profiles with the bayesian network information in XML files.

Example Screenshots of Sports4U The sports news ticker presentation can be viewed, e. g., with a SMIL player over the web as shown in Figure 10.8. The figure depicts two example screenshots of sports event presentations presented by our personalized sports news ticker. The screenshot on the left hand side shows a sports event presentation in a template employing an image element and text element. The right hand side screenshot depicts a sports event presentation using a series of image elements together with a corresponding audio record.

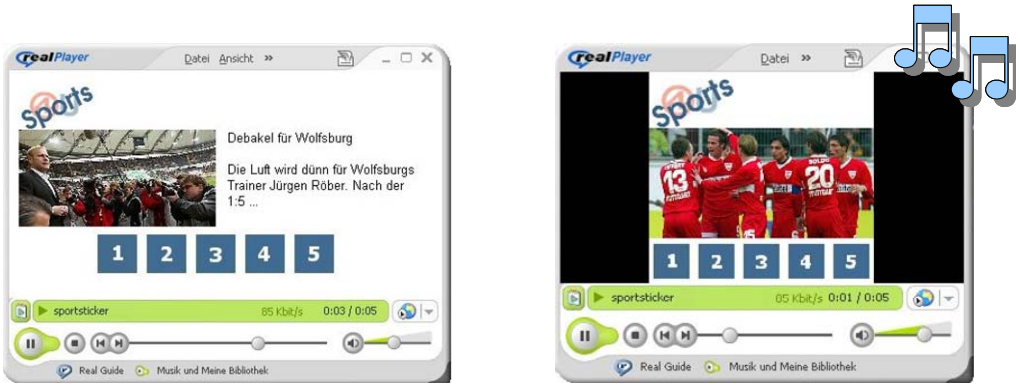


Figure 10.8: Screenshots of the personalized sports news ticker Sports4U

To view the personalized sports news ticker, the users need to register for the Sports4U application. When registering, the users initially state their sports interests in regard of the different kinds and categories of sports they like (see above). In addition, the users can define their preferences in regard of, e. g., the presentation's style and background color, language, and maximum duration of the sports news ticker. Once, the users have registered and initially stated their sports interests, the Sports4U application selects the most appropriate sports events for presenting them to the users.

Relevance Feedback When presenting the selected sports events, Sports4U automatically adapts the information and assumptions it has about the users' sports interests by providing a relevance feedback functionality. This relevance feedback functionality allows the users to valuate the personalized selection of the sport news events by marking those sports events that are more interesting than others. This valuation of the news selection is shown in Figure 10.8 by the five boxes with the numbers from one to five. The users can click on these numbers to valuate the relevance of the currently presented sports event. A value of one indicates the Sports4U application that the currently presented sports event is very relevant to the user. On the other end of the scale, a value of five represents that the user is not interested at all in the currently presented sports event. Consequently, a value of three represents a neutral valuation of the presented event.

On the basis of the relevance feedback the users provide, the Sports4U application adapts its assumptions about the users' sports interests. For example, if a user valuates a specific sports event as not interesting, it is very likely that he or she does not want to receive similar news in future. On the other hand, sports events in the MediÆther associated to sports categories that are valued as very relevant should be more added to the user's ticker. The Sports4U application gradually adapts the user's profile information to include more relevant sports events and to exclude those who the user has valuated as uninteresting. Consequently, the valua-

tion yields in a modified selection and composition of sports events for the individual user's ticker.

Summary The Sports4U application allows for an automatic selection and presentation of sports events according to the user profile information. Employing a relevance feedback functionality, the application updates the information and assumptions it has about the users' sports interests according to the provided valuation of the selection results. In a future extension of our work, we like to abstract from the relevance feedback functionality provided by the Sports4U application and aim at integrating an abstract relevance feedback component into the MM4U framework. The personalized sports news ticker provided by the Sports4U application can be, e. g., embedded into a web site of a sports portal. Here, the users can continuously be updated with relevant and interesting sports news. When new sports events are added to the MediÆther, the personalized ticker is updated accordingly during runtime and thus the users burden of actively watching and searching for new and relevant sports events is alleviated.

10.3 Manual4U—Interactive Instruction Manuals on Mobile Devices

- Name: Manual4U
- Application area: mobile instruction manuals
- Usage of the MM4U Framework: extension of the internal multimedia content representation model by composition operators for form functionality, AWT-based multimedia player for the internal multimedia document model
- Impact: learning of procedural actions with mobile devices

With Manual4U, a flexible application is developed that aims at providing instruction manuals for different mobile devices. These mobile instruction manuals enable mobile users to gain knowledge in how to conduct specific procedural actions. The instruction manuals shall not be developed by plain programming them as this requires high effort, especially when adapting them to different mobile end devices. To provide non-programmers support for creating complex, interactive instruction manuals, the Manual4U application provides a generic and flexible instruction manual definition language (IMDL) based on XML. The aim of this IMDL is to provide for the write-once-run-everywhere principle, i. e., to write once the instruction manuals in the declarative definition language and then to execute this manual on different mobile end devices.

Definition of Mobile Instruction Manuals An instruction manual in the Manual4U application consists of a set of single instruction steps that are to be conducted. For each step, the required media elements are specified. For it, different sets of step templates can be defined with the IMDL for different end devices. The Manual4U application predefines step templates that employ different media types. For example, there exists a step template that uses a single image element only, a template that supports a long textual description, a template with a short textual description

and an image element, as well as different step types including video and audio elements. For the Manual4U application, we defined two sets of these step templates. One set is targeted for PDAs such as Pocket PCs and another set is aimed at cell phones such as Sony Ericsson P800/P900/P910(i). These device specific sets of step templates are employed for the rendering of different instruction manuals on the targeted mobile end devices.

Besides defining the single steps of an instruction manual by employing the different step templates, it is important to define the order in which these steps can or must be conducted. The order of conducting the steps is important to define as there can be some interdependencies between them. For example, for employing a specific electronic device, first the power supply must be connected and the device possibly needs to be configured accordingly before it can be used actually. Thus, the IMDL allows for defining for each step in the instruction manual, when the step can be conducted, i. e., which preconditions exist and which steps need to be conducted first. With defining such step interdependencies, the Manual4U application can automatically determine when to activate and deactivate a specific step in the interactive instruction manual. This allows the Manual4U application for guiding the users through the interactive instruction manual. By this, the Manual4U application supports the users to learn how to conduct a specific procedure defined in the manual.

So far, the different steps of an instruction manual as well as their interdependencies are defined. For actually rendering the interactive instruction manual, the single steps need to be arranged on different manual pages. This page layouting task is typically left over to the Manual4U application to automatically arrange the steps on instruction manual pages for the different end devices. However, it can also be conducted manually in the instruction manual description, e. g., to force a page break at a specific step.

Technical Foundation and Implementation The technical foundation for developing the Manual4U application forms the mobileMM4U framework. Here, the instances of the three components User Profile Accessor and Connectors, Media Pool Accessor and Connectors, and Multimedia Composition of the mobileMM4U framework are employed. Besides these framework components, a new component has been developed for the Manual4U application. This newly developed component is called InternalModelPlayer and is aimed at taking multimedia content specified in the internal representation model as input in order to directly render it on the user's end device. Thus, it is a multimedia player for our internal multimedia content representation model.

For the User Profile Accessor and Connectors component, we employ the FilesystemUserProfileConnector the mobileMM4U framework provides (see Section 8.16). By this connector, the Manual4U application extracts the user's preferences in regard of presenting the interactive instruction manual. This includes the preferred language, media type, and length of text. In regard of the Media Pool Accessor and Connectors component, a specific extension of the FilesystemMediaElementsConnector has been developed. Within this connector, a medium element consists of a set of different pre-generated media resources. Each medium resource within this set describes the same medium element, however, with different concrete attributes. For example, the media resources of an image element can differ, e. g., in regard of

their resolution, format, and spatial dimensions. Text elements can be described by media resources that differ in attributes such as language and length of their description. Media resources for audio elements and video elements take into consideration the language, quality, and bandwidth. The media resources of the single media elements are described in appropriate XML files. When the Manual4U application requests a specific medium element, it employs the query object the Media Pool Accessor and Connectors component provides (see Section 8.7.1). Together with the user profile information, the query object is used by the Manual4U application to retrieve for the requested medium element that medium resource from the media elements connector that best matches the user's preferences and end device characteristics. The user's preferences include, e. g., the preferred language, whether short or long textual descriptions shall be selected, and whether image elements shall be received together with text elements or if the textual descriptions are to be replaced by audio files. In regard of taking the different mobile end devices' characteristics and capabilities into account, the media elements are provided in different resolutions, color depths, and file formats.

For the composition of the instruction manuals, the internal multimedia content representation model defined by our Multimedia Composition component instance is employed. However, for the Manual4U application the internal multimedia content representation model is extended by basic composition operators for forms functionality. These form composition operators provide abstractions of typical form elements functionality such as they are provided by HTML and XHTML. Examples of these form elements are (submit) buttons, radio buttons, check boxes, and input fields. The corresponding form composition operators in our internal representation model are, e. g., Button, CheckBox, RadioButton, and InputField. Thus, the interactive mobile instruction manuals are authored in an extended version of the internal multimedia content representation model.

For rendering the interactive instruction manuals in the extended representation model on the mobile devices, the InternalModelPlayer component introduced above is employed. The implementation of the InternalModelPlayer component bases on the Abstract Window Toolkit (AWT) of Java [Sun06]. The communication between the form composition operators of the AWT-based rendering of the instruction manual and the underlying Manual4U application logic is conducted by employing Java's listener concept for graphical user interfaces (as it is provided by Java AWT and Java Swing [Sun06]).

Example Screenshots of Manual4U Figures 10.9 and 10.10 show example screenshots of the Manual4U application running on a Pocket PC and a Sony Ericcson P910i, respectively. In general, the screen of the Manual4U application is divided into two parts. The bigger upper part presents the actual mobile instruction manual with the single steps to be conducted. This part is rendered by the AWT-based multimedia player component. The smaller lower part contains four page-navigation control buttons of the Manual4U application. These buttons allow the mobile users to switch between the single pages of an instruction manual and to skip directly to the first and the last page of the manual, respectively. The page-navigation control buttons are provided by the Manual4U application logic.

The screenshots in Figure 10.9 depict an interactive instruction manual for baking a tuna pizza. Whereas Figure 10.9(a) shows the Manual4U application running

on a Pocket PC for a user preferring German, Figure 10.9(b) is taken from a Sony Ericcson P910i targeted at a user preferring English. The interdependencies of the single steps are shown, e. g., in Figure 10.9(a). Steps that are already conducted are marked by the users with a tick (the upper step 3). The next steps that can be conducted are indicated with an activated empty box (the step 4 in the middle). Once, the users have successfully conducted such a step, they can mark it like the upper one. Steps that require further prerequisites to be fulfilled are indicated with an inactive and thus shaded box (the lower step 5). These steps cannot be marked as finished by the users until the necessary prerequisites are fulfilled. As the screenshots in Figure 10.9 show, the layout of the manual's pages is different for the employed end devices, as the available spatial dimension of the display is different.

(a) (b)

Figure 10.9: Screenshots of a mobile instruction manual for baking tuna pizza

Figure 10.10 shows screenshots of an interactive instruction manual for learning how to use a pipette in a laboratory. Again, the screenshot shown in the left hand side is taken from a Pocket PC and is targeted at a user preferring German and the right hand side screenshot is captured from a Sony Ericcson P910i for a user with English as language preference. Whereas for the user of the left hand side image elements as well as text elements are selected for presenting the instruction manual, the screenshot on the right hand sight shows the same instruction manual employing an image element together with an auditive narration.

Summary The presented Manual4U application allows mobile users to learn how to conduct specific procedures defined by the corresponding interactive instruction manuals. With supporting for defining step interdependencies, the Manual4U application guides the users through the instruction manuals and shows, when which steps of the instruction manual can be conducted. By marking the steps conducted by the users with a tick as shown in Figure 10.9, the users can keep track of the activities of the instruction manuals conducted so far.

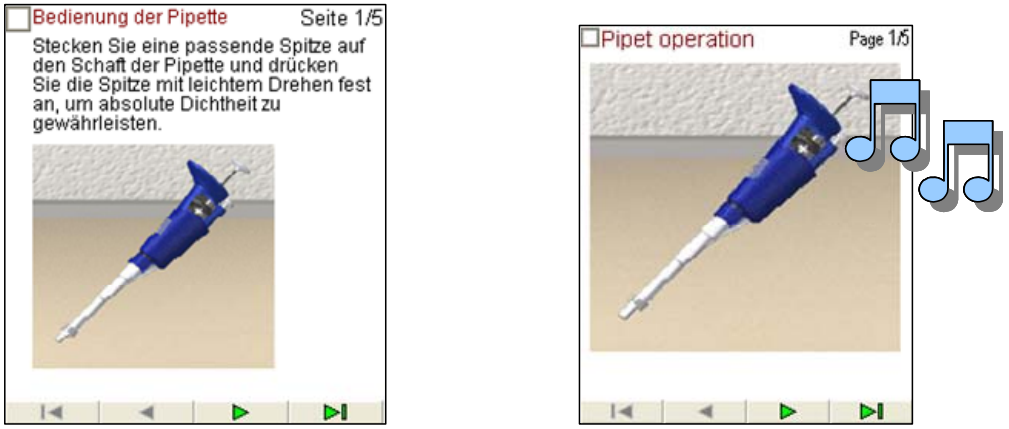


Figure 10.10: Screenshots of an instruction manual for using a laboratory pipette

10.4 Transformation4U—A Multi-channel Multimedia Presentation Generation Service

- Name: Transformation4U
- Application area: personalized multimedia applications that need to provide for a multi-channel delivery of multimedia content
- Usage of the MM4U Framework: uses the Presentation Format Generators component only
- Impact: service provider for transforming multimedia content

Our multi-channel approach for creating personalized multimedia presentations in different presentation formats targeted for different (mobile) end devices is not only embedded within the MM4U component framework. However, this multi-channel generation of multimedia content provided by the Presentation Format Generators component can also be seen as service provider for multimedia applications to bring their content over the last mile to the end user's device setting. This is the goal of our Transformation4U service. The service takes as input multimedia content delivered in a XML-based encoding of our internal multimedia content representation model. Then, the Transformation4U service internally employs the Presentation Format Generators component instance of the MM4U framework to create the multimedia presentation in the final format. Thus, the technical basis of the Transformation4U service is based on the Presentation Format Generators component only.

The Transformation4U service is provided in form of a Java servlet running on a Unix server with the Apache Tomcat web server [The06a]. For this servlet, we set up a web page describing how to apply the multi-channel Transformation4U service and how to integrate it into concrete multimedia applications. For using the Transformation4U servlet to deliver multimedia content in different presentation formats, the following five steps are performed.

1. The multimedia application needs to create the multimedia content according to the DTD of the internal multimedia content representation model. This DTD is presented in Appendix C.
2. Then, the multimedia application sends the multimedia content in form of a XML document as first parameter to the Transformation4U servlet. In addition, the application needs to specify the targeted presentation format as second parameter.
3. The Transformation4U servlet deserializes the multimedia content provided as a XML document and creates all required objects of the internal multimedia content representation model. Then, the Presentation Format Generators component instance of the MM4U framework is employed to transform the multimedia content to the targeted presentation format.
4. The Transformation4U servlet returns the multimedia presentation in the final format to the calling application.
5. The application displays the presentation in the final format to the user.

A concrete application employing our Transformation4U service is the multimedia database METIS, developed at the Austrian Research Studio for Digital Memory Engineering [RSA06]. The METIS multimedia database is briefly described in Section 8.7.3.4. Here, also the METISMediaElementsConnector is presented that provides the MM4U framework access to media elements stored in the METIS database. With the METISMediaElementsConnector, we integrated the METIS database into the MM4U framework. With the Transformation4U service, the integration is inversed. As depicted in Figure 10.11, the METIS database has integrated our Transformation4U service to provide multimedia documents stored in the database in different presentation formats. This integration is realized by developing an appropriate Transformation4U plug-in for the METIS database. The plug-in provides XML-based multimedia presentation templates following the DTD of the internal representation model of the Multimedia Composition component. These templates are stored in the METIS database and contain references to media elements that are also stored in the METIS database. The METIS templates are send by the plug-in to the Transformation4U service, which transforms the multimedia content and returns a presentation in the final presentation format.

The templates provided by the METIS database are currently rather static. Thus, in a future extension the MM4U framework could be integrated into the METIS database such that the METIS templates are dynamically generated by the MM4U framework. Here, placeholders for media elements would be defined in the METIS templates that carry query information for dynamically determining the actual media elements. These placeholders would be filled in by the MM4U framework with (complex) media elements from the METIS database. Consequently, such dynamically generated METIS templates would exploit the integration of the MM4U framework and METIS database in both directions. The MM4U framework uses the METIS database integrated via the METISMediaElementsConnector for determining and filling in the METIS template's placeholders with concrete media elements. In the next step, the METIS database employs the Transformation4U service for actually deploying the dynamically authored multimedia content into different presentation formats.

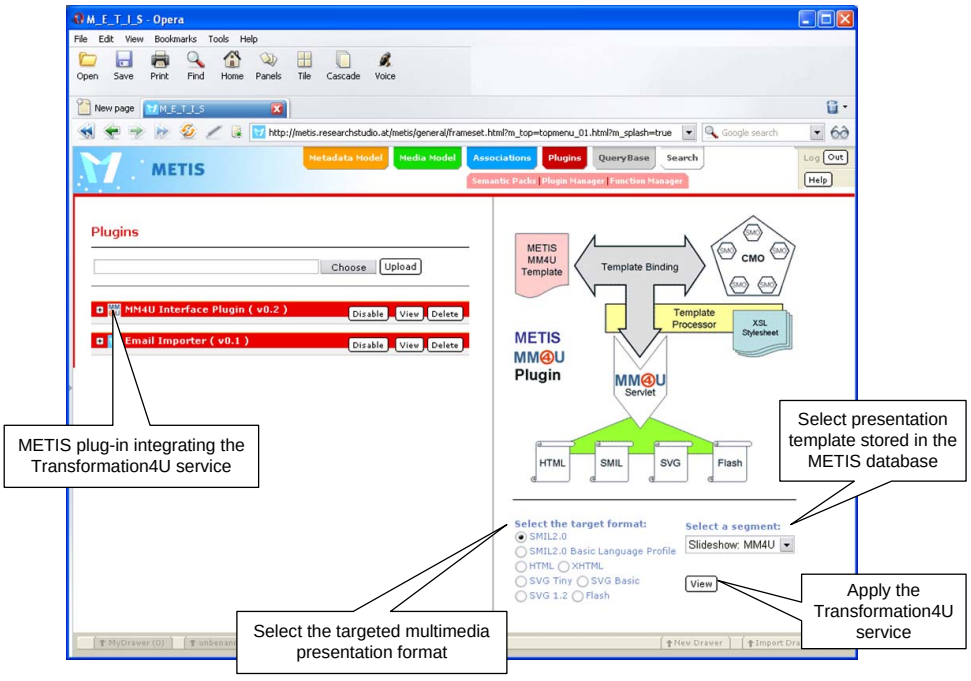


Figure 10.11: Screenshot of the METIS multimedia database employing the Transformation4U service

10.5 Pictures4U—A Personalized Picture Albums Generator

- Name: Pictures4U
- Application area: digital photo services
- Usage of the MM4U Framework: all components of the MM4U framework; employs the media elements connector to the multimedia database METIS
- Impact: creation of personalized photo albums

With digital cameras, huge amounts of personal pictures are taken, e. g., at conferences, summer vacation, family events, parties, and festivities. These many collections of pictures need to be managed. The aim of the Pictures4U application is to provide a personalized access to these pictures as it is often needed to, e. g., create a presentation of a “Friend’s Birthday Party”, a “Farewell Gift for a Colleague”, or to present your “Last Year’s Summer Vacation Highlights”.

With the Pictures4U application, we provide a web-based service for automatically authoring and delivering personalized photo albums. A photo album consists of a series of photos stored in the Pictures4U server. Each photo can be described by a title and optionally has a longer textual description. An optional start screen can be determined for each photo album. This start screen can be used, e. g., to show

the subject of the photo album such as a title for the summer vacation album or the name of the friend at which birthday's party the photos were taken. In addition, an individual background music can be determined for the photo albums. When the photo album is presented to the users, each photo is presented one by one. In addition, the provider of the photos can specify for each photo album the presentation parameters such as the default presentation duration for each photo, if navigation buttons allowing for interactively navigating within the slides of the presentation shall be shown, or if a background music shall be played while presenting the photos.

For the viewers, i. e., for the users of the Pictures4U application the generated photo albums can be personalized in regard of the preferred presentation format and the characteristics of the used end device of the users. In addition, the user's preferences in regard of different selection and presentation criteria of the photos can be taken into account when authoring the personalized photo albums. For example, a specific user may not like the default presentation duration of, e. g., four seconds for each photo but likes to watch the photos for a longer time period. Thus, the default duration determined by the provider of the photo album is overridden and the user's preference is used instead. The same holds for the navigation buttons. If the users have a specific preference here, the default value provided by the author of the photo album is ignored and the photo album is presented either with navigation buttons or as a non-interactive slideshow. In addition, for users with time constraints the total number of photos can be reduced to a specific number. To choose which photos of an album are omitted for the presentation, different parameters are applied. These parameters are, e. g., the exposure and similarity of the photos, and are provided by the URIMediaElementsConnector presented in Section 8.7.3.1. For example, photos that are overexposed or underexposed as well as very similar photos according to their histogram are omitted first. In addition, time clustering information of the photos are taken into account to determine photos that form distinct time clusters and thus are likely to present the same event. Depending on how many photos a specific time cluster contains, only a selection of these photos is included into the actual presentation. This time clustering functionality is employed to realize the reduction of the number of photos shown in a presentation to a specified maximum number. For storing and managing the user profile information a solution on the client side is employed for the Pictures4U application. We use a CookieUserProfileConnector integrated in our User Profile Accessor and Connectors component instance for storing the user profile information as cookies within the user's web browser.

Besides the URIMediaElementsConnector, also a second variant of the Pictures4U application has been developed. This variant employs the multimedia database METIS and the METISMediaElementsConnector for access to the media elements (see Section 8.7.3.4). Here, an appropriate import procedure has been defined to bring the photos into the METIS database. First, the meta data of the photos such as exposure, similarity, and time clustering information are determined, which are necessary for the subsequent selection of the photos for the album presentation. In a second step, the photos with these automatically derived meta data together with the manually added title and description are loaded into the METIS database for storage.

Figure 10.12 depicts some screenshots of our Pictures4U application. The screenshot on the left hand side shows the web page that allows the users for selecting a photo album they like to watch as well as to determine their presentation preferences. On the right hand side, a sample screenshot of a photo album in MPEG-4 format about "Our Vacation on Island Norderney" is shown. The photo album is targeted at a Desktop PC and employs the user's preference in regard of showing each photo for ten seconds, playing the background music, and disabling navigational interaction within the album.

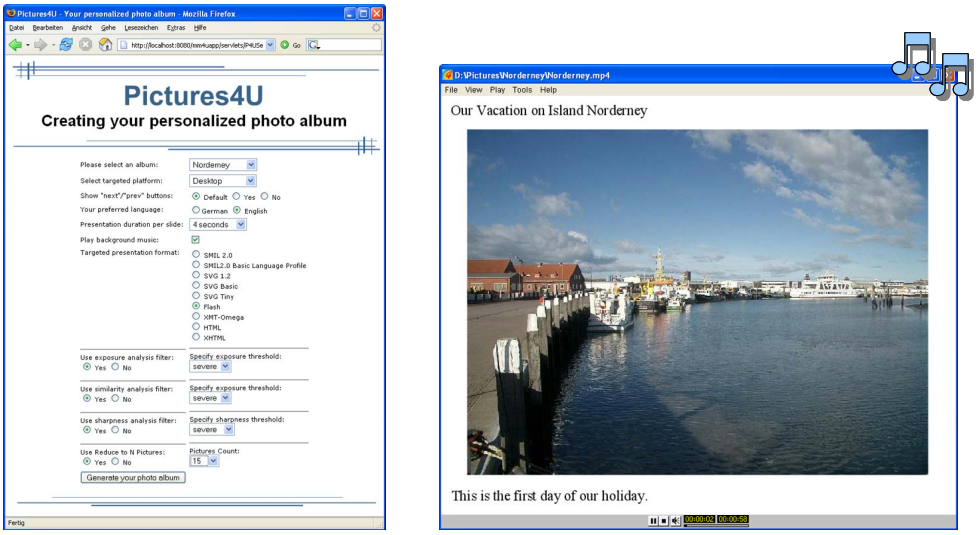


Figure 10.12: Screenshots of the Pictures4U application showing the album selection web page and a sample photo album presentation

The presented Pictures4U application aims at presenting personalized photo albums in the Internet. It allows for a flexible playback of the dynamically authored photo albums according to the user's preferences. Future extensions of the Pictures4U application could be to provide support for community building features. Here, the provider of a photo album could determine which users are allowed to watch specific photo albums. In addition, the providers of a photo album could determine whether making comments shall be provided or if further sharing of the album to the viewer's friends shall be allowed. For providers of the photo albums, a web-based upload mechanism could be developed that allows for an easy integration of new photo albums.

10.6 *x*SMART—Context-driven Smart Authoring of Multimedia Content

- Name: *x*SMART
- Application area: context-driven creation of multimedia content

- Usage of the MM4U Framework: interactive graphical manipulation of multimedia content in the internal representation model, abstraction from the MM4U framework for semi-automatic multimedia authoring support
- Impact: creation of multimedia content by domain experts

In recent years, many highly sophisticated multimedia authoring tools have been developed (see Section 3.2.2). Up to today, these system's integration of the targeted user context, however, is limited. With the *Context-aware Smart Multimedia Authoring Tool* (*x*SMART), we developed a semi-automatic authoring tool that integrates the targeted user context into the different authoring steps and exploits this context to guide the author through the content authoring process [SB05b]. The design of *x*SMART allows that it can be extended and customized to the requirements of a specific domain by domain-specific wizards. These wizards realize the user interface that best meets the domain-specific requirements and effectively supports the domain experts in creating their content targeted at a specific user context.

Multimedia content authoring is the process in which synchronized multimedia presentations composed of different discrete and continuous media elements are created (see Section 3.1.3). The authoring of multimedia content is a challenging task and typically needs a high expertise in both the domain and often also in using the authoring tool, too (see Section 3.2.2). Multimedia authoring becomes even more challenging when the created multimedia presentation needs to meet different devices, platforms, and multimedia formats and at the same time the high demand for personalization of content for the individual user or user group [BH05]. Then, only the relevant media elements, the proper media formats, and the right modality need to be chosen such that the end users finally receive a presentation that meets their personal interest, the technical infrastructure, and (mobile) end device.

The field of multimedia authoring produced a number of tools ranging from domain expert tools to general purpose authoring tools. However, multimedia authoring is still an open issue, as these tools are still tedious to handle and not practical for the domain experts. To contribute to a less-sumptuous authoring process that is aware of the targeted user context, we developed *x*SMART.

This wizard-based tool integrates the user's context into all steps of the authoring process from the content selection, content composition to the creation of the multimedia presentations in the final presentation formats. It provides support for a semi-automatic multimedia composition in cases in which an automated creation of multimedia content is not desired, e. g., a very specific content domain in which an expert is still wanted for the decision which content to compose. However, we did not build yet another authoring tool but designed it such that it can be used and extended by arbitrary domain-specific wizards. Basing on the MM4U component framework, the *x*SMART tool exploits several of the framework provided components.

In the following Section 10.6.1, the context-driven authoring of multimedia content is described. Then, the general architecture of the *x*SMART tool is presented in Section 10.6.2. Finally, in Section 10.6.3 the implementation of the authoring tool is described and a concrete wizard for the domain of personalized photo albums is presented in Section 10.6.4.

10.6.1 Context-driven Authoring of Multimedia Content

The general context-driven authoring process of multimedia content and the integration of the targeted user context is depicted in Figure 10.13. It adopts the general multimedia authoring process as presented in Section 5.1 to a (semi-)automatic, context-driven creation of personalized multimedia content. In the selection phase, the targeted user context can be used to narrow down the different options for the multimedia composition. Hence, the authoring environment presents only those media elements for an inclusion into the presentation that are technically and/or semantically relevant for the presentation. For the composition, the context plays the role of determining the style and layout of the presentation. For transforming the assembled multimedia content to concrete presentation formats only those output formats are offered, that are reasonable for the selected (mobile) end device. Hence, the context stands by to relieve the author from decisions and tasks during the authoring process which can be "foreseen" by the targeted user context and, hence, are hidden from the author. What type of context information is actually needed and exploited to support the authoring process depends on the concrete application domain. It can be, e. g., the targeted end device or multimedia player, the user's preferred presentation style and language, as well as any other domain-specific context information.

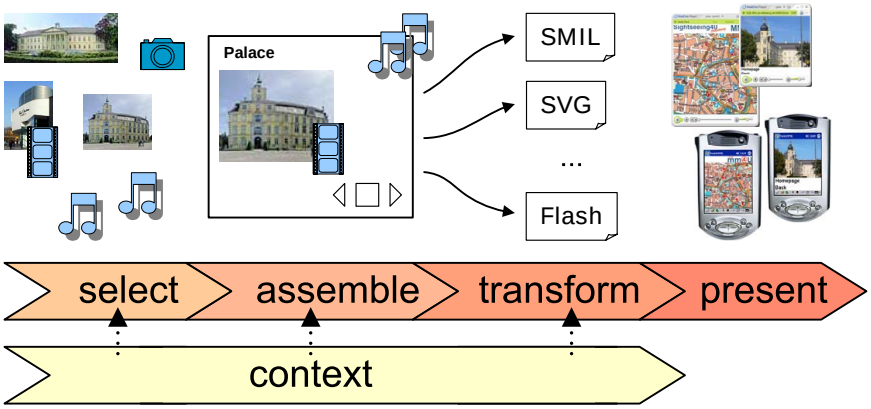


Figure 10.13: General process chain for the context-driven authoring of personalized multimedia content

Domains in which such a semi-automatic context-driven authoring of personalized multimedia presentation is desired are, e. g., in the e-learning context. Here, one can imagine an author that designs a multimedia course in cardiac surgery and is supported in the different authoring steps for generating a personalized course for a targeted audience. Another possible application domain is the creation of personalized photo albums. Here, the authoring tool would guide the users through the creation process to create a photo album about, e. g., "Highlights of the Summer Vacation" or "My Sweet Kids", that can also be targeted at a specific user or user group, e. g., for friends or the grandparents.

10.6.2 Architecture of xSMART

With the xSMART authoring tool, we aim at supporting the development of authoring wizards that will be employed by domain experts for the multimedia content authoring. Therefore, the xSMART architecture provides an abstraction of general purpose authoring tools and offers only the very basic set of multimedia composition functionality. This includes selecting media elements and their interactive composition in a page-oriented fashion. In addition, the xSMART authoring tool allows for the definition and employment of presentation templates for the different pages. So far, the system is not different to existing approaches in the field. However, the design differentiates a concrete authoring wizard from basic authoring features, which again abstract from the concrete composition of a multimedia document.

Figure 10.14 illustrates these different levels of multimedia authoring: The basis of this architecture forms the MM4U component framework. On top of it, the xSMART authoring tool adds another level of abstraction with suitable interfaces, targeted at a simplified multimedia authoring. Using the xSMART authoring tool, domain-specific wizards can be developed to support domain-experts in their context-driven multimedia content creation task as presented in Figure 10.13. These domain-specific wizards can be plugged into the xSMART tool by employing component technology. The wizards guide the domain expert step-by-step through the context-driven authoring process, while hiding away the technical details of the authoring process. This enables our xSMART authoring tool to best meet the requirements and needs of the domain experts, i. e., to effectively and efficiently support the domain-experts in their content creation task and hide as much technical details of the authoring process from the users as possible. In contrast with the other applications, which are employing the MM4U framework for the *automatic* generation of personalized multimedia content, our xSMART authoring tool provides support for the development of context-driven semi-automatic multimedia authoring wizards.

10.6.3 Implementation of xSMART

We use the MM4U framework as basis to build the multimedia composition and personalization functionality of the smart-authoring tool xSMART. For this, the Multimedia Composition component supports the creation and processing of arbitrary document structures and templates. The authoring tool exploits this functionality for composition to achieve a document structure that is suitable just for that content domain and the targeted audience. The Media Pool Accessor and Connectors component supports the authoring tool in those parts in which it lets the author choose from only those media elements that are suitable for the intended user contexts and that can be adapted to the user's infrastructure. Using the Presentation Format Generators component, the authoring tool finally generates the presentations for the different end devices of the targeted users. Thus, the authoring process is guided and specialized with regard to selecting and composing personalized multimedia content. For the development of this authoring tool, the framework fulfills the same function in the process of creating personalized multimedia content like with the multimedia applications described in the previous sections. However, the

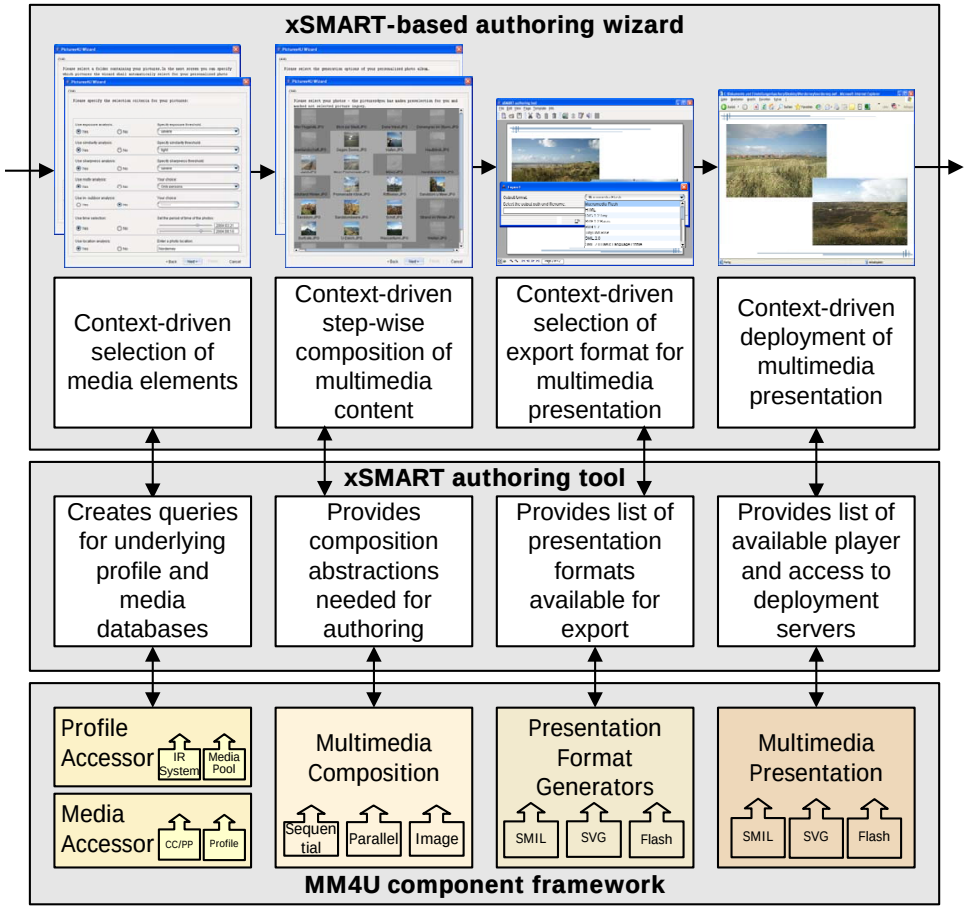


Figure 10.14: Architecture of the context-driven smart authoring tool *x*SMART

creation of personalized content is not achieved at once but step by step during the authoring process.

Figure 10.15 shows a screenshot of the interactive authoring tool *x*SMART and its central multimedia content authoring interface. As shown in the figure, different media elements such as image, text, audio, and video elements can be added and arranged in the page-oriented multimedia presentation. Pages can be added, removed, and resorted within the presentation. The buttons on the bottom left of the authoring tool allow for changing the presentation's pages as well as switching to the template mode. To allow for interactively composing and manipulating the multimedia content via the provided graphical user interface, the design pattern model-view-controller [GHJV04] is employed for implementing the *x*SMART tool. The controller and view are realized by applying Java's Swing library [Sun06]. The model constitutes the abstract multimedia content representation model of the Multimedia Composition component. Consequently, the composition and manipula-

tion of the multimedia content within the *x*SMART tool is directly conducted on the MM4U framework's internal multimedia content representation model. Thus, manipulating the page-oriented multimedia content authored with the *x*SMART tool means to directly work on and manipulate the content's representation in the internal composition model.

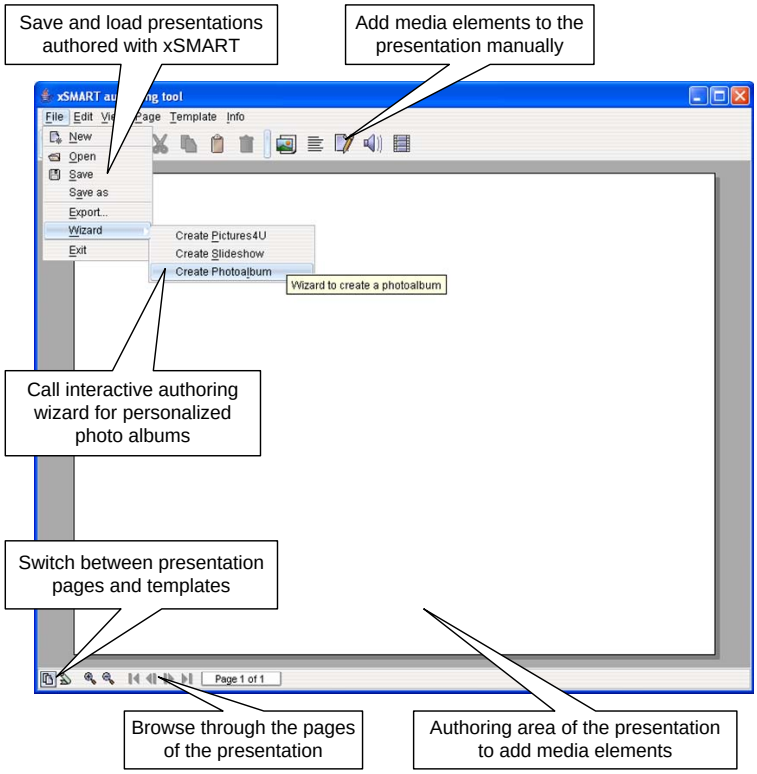


Figure 10.15: Authoring of page-oriented presentations with the smart authoring tool

10.6.4 Example of a Context-driven Smart Authoring Wizard for *x*SMART

So far, an authoring tool has been presented with *x*SMART that abstracts from the general purpose authoring tools as presented in Section 3.2.2.2. It provides a very basic set of multimedia composition functionality. In this section, we present an example of extending the *x*SMART authoring tool by a domain-specific authoring wizard in the domain of personalized photo albums. This photo album wizard conceptually bases on the Pictures4U application presented in Section 10.5. Following the general process chain for the context-driven authoring of personalized multime-

dia content presented in Section 10.6.1, the following steps are typically conducted by such a domain-specific authoring wizard.

Selection First, the media elements are selected that are needed for the multimedia presentation. Here, the wizard supports the author in preselecting media elements for the considered domain and according to the targeted user's or user group's context. For example, with our wizard for creating personalized photo albums, an author can define that only pictures taken during summer time or any other specific time are selected. In addition, the users can determine that only photos with a good exposure as well as sharpness are chosen or that the album shall employ only photos taken indoor or outdoor, respectively. Therefore, the *x*SMART authoring tool creates queries for the underlying profile and media databases of the MM4U framework and passes the results back to the domain-specific wizard.

The personalized photo album wizard consists of a set of authoring pages that guide the users through the authoring process of creating a personalized photo album. Figure 10.16 shows the screenshots of the photo album wizard's support for providing the selection of the pictures. The first page of the wizard is depicted in Figure 10.16(a). It allows the users to determine the directory in which the photos for the album are stored. On the second page of the wizard shown in Figure 10.16(b), the users specify the parameters the wizard shall employ for determining which photos of the selected directory are actually to be included into the photo album. Once the users switch to the third page of the wizard by clicking on the "next"-button, the photos of the directory are presented. As shown in Figure 10.16(c), those pictures that fulfill the specified parameter settings are highlighted. The pictures that did not pass the selection criteria are shaded. These pictures will not be employed for the composition of the photo album. By clicking on the shaded images, the users can manually add pictures to their album. In addition, by clicking on pictures that passed the selection criteria, the users can manually remove those images from the photo album.

Composition and Preview For the composition of media elements, *x*SMART defines some useful abstractions of the fine-granular multimedia composition functionalities the MM4U framework provides. For example, a medium element presented on the user interface is internally represented in *x*SMART by an internal model's medium element, an appropriate projector, and a temporal selector. In addition, it can optionally comprise a link operator. These abstractions allow an authoring wizard for an as simple as possible composition of media elements and links in time and space in a page-oriented multimedia presentation. However, as these abstractions still will be too tedious to handle for the user creating a personalized photo album, our wizard hides these authoring details from the user. As shown in Figure 10.16(b), the users only need to specify some selection parameters which are used as input for the composition step. Having selected the photos that shall be used for the personalized album as shown in Figure 10.16(c), the users can specify further composition parameters such as the targeted (mobile) end device, the page layout, and the photo album's style to be used. Having selected for a specific end device, the *x*SMART tool provides for automatically switching to the best suitable presentation dimension

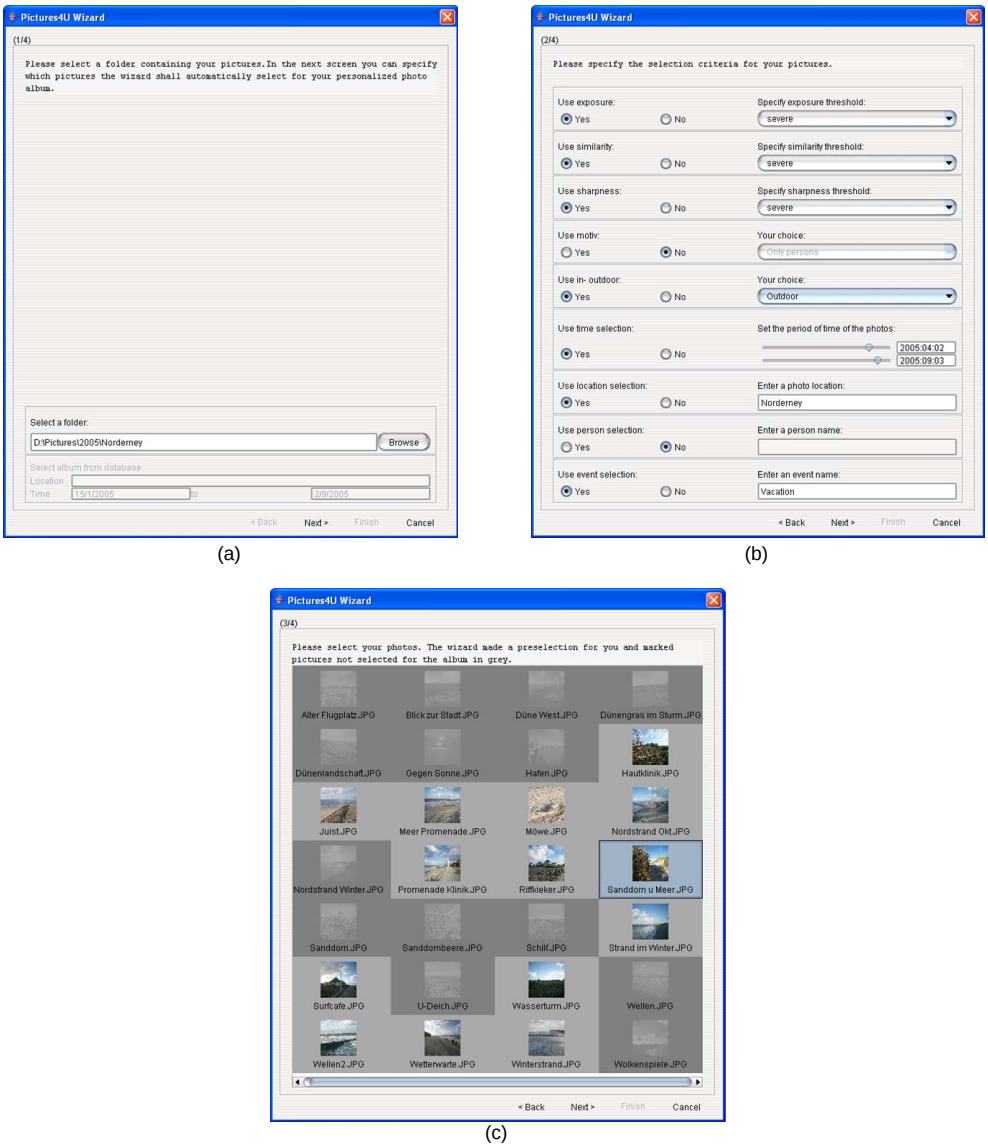


Figure 10.16: Screenshots of the photo album wizard's support for selecting photos

for composing the media elements. The wizard's page for setting these composition parameters are depicted by the screenshot in Figure 10.17.

When the users click on the "finish"-button, the wizard actually creates the pages of the personalized photo album. Subsequently, the users can preview the authored multimedia content as shown in Figure 10.18 within *x*SMART's main au-

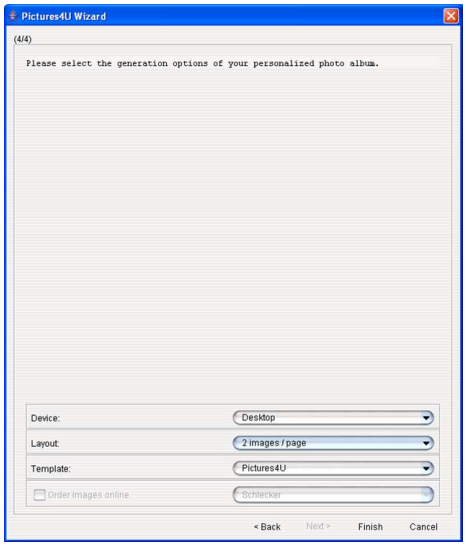


Figure 10.17: Screenshot of the photo album wizard's page for selecting the composition parameters

thoring screen. Here, all details of the multimedia composition can be changed and modified. This means for the photo album to, e. g., switch album's pages, adding and removing image elements or text elements, changing the layout and the used presentation template, and the like.

Export and Deployment When finishing with the preview of the multimedia presentation, the users can export the authored multimedia content. For it, the users employ a second wizard of the *x*SMART tool. This export wizard provides a list of concrete presentation formats that are supported by the underlying MM4U framework and which are supported by the selected end device. A screenshot of this export wizard is depicted in Figure 10.19. Once selected a target format and determined a filename for the photo album, the multimedia content is exported and the transformation result can be viewed in an appropriate multimedia player. Figure 10.20 shows our example presentation exported to the Flash format and presented by Adobe's Flash plug-in. Finally, the exported multimedia presentation can be distributed, e. g., by uploading it onto a web server or sending it by e-mail. In a future extension of our context-driven wizard for the semi-automatic authoring of personalized photo albums, we aim to provide a service to order a physical print of the just created photo album online.

10.6.5 Summary

We motivated the need for a context-driven multimedia authoring. The presented context-aware smart multimedia authoring tool can be seen as first step towards

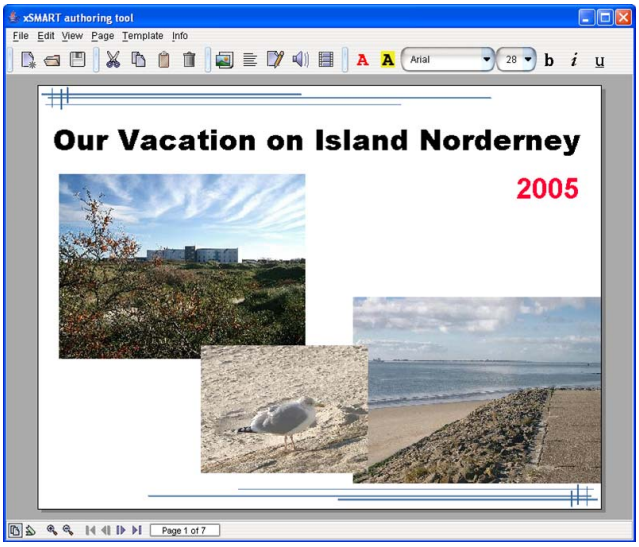


Figure 10.18: Screenshot of the preview of a personalized photo album

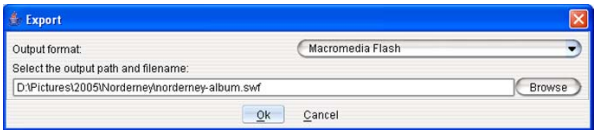


Figure 10.19: Screenshot of the export wizard for the personalized photo albums

a sophisticated authoring suite of context-driven multimedia content authoring. It forms a common basis to develop application-specific plug-ins and wizards, providing support for specialized multimedia authoring tasks. With *x*SMART and the underlying MM4U framework, we contribute to a more efficient and economic development of domain-specific tools for multimedia authoring. Such domain-specific authoring wizards allow the users for a quick and easy creation of multimedia content by employing context as an intuitive means to narrowing down the media elements selection task and facilitating the multimedia composition task.

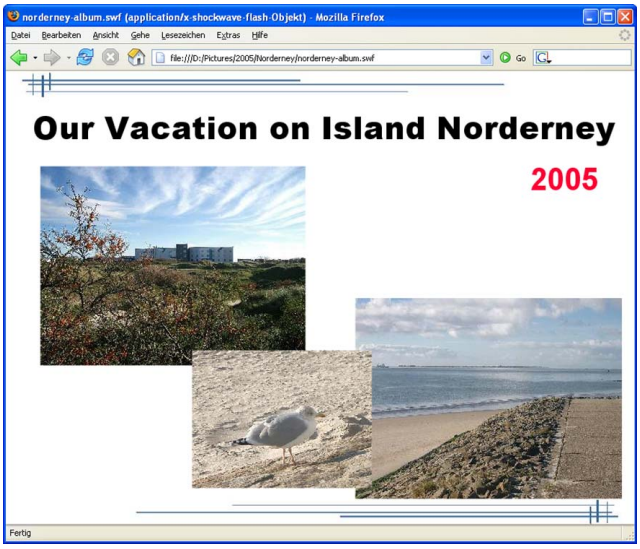


Figure 10.20: Screenshot of an exported photo album in the Flash format

11 Lessons Learned

With the canonical process of media production [Har05], Hardman claims for a more integrated support for multimedia authoring and a better understanding of the media production process. The MM4U framework components define distinct interfaces, unified data structures, and services needed to author personalized multimedia content. By this, we provide for a better understanding and integration of the single tasks and different phases involved in the media production process. A specific functionality such as a media connector for a particular media storage solution can be encapsulated by the corresponding framework component. Access to such a media connector is only provided by the contractually-specified component services. Thus, a component-based approach provides for a better exchangeability of the functionality by means of replacing and exchanging component instances. In addition, as the components of the MM4U framework are realized as traditional object-oriented frameworks, the framework components shield these object-oriented frameworks from the outside world. Consequently, with the ProMoCF approach, we allow for composing different object-oriented frameworks by means of component technology. With this approach, we avoid the problem of fragile base classes that used to arise when combining object-oriented frameworks (see Section 4.1.2.1).

The MM4U framework can be used as a whole. However, the components can also be independently employed. This allows the multimedia research community to apply the MM4U framework and its single components to support and realize their multimedia content creation system in a (possibly more) specific topic in the field of multimedia engineering and thus to provide for a better understanding of the media production process. The MM4U framework defines a generic component architecture for personalized multimedia applications that can be directly applied for the development of such applications. In addition, it is adaptable and extensible in regard of the requirements of a concrete application. Thus, the MM4U component framework provides for the overall goal of a more efficient and economic development of personalized multimedia applications (see process improvement requirements in Section 5.2.3). The seamless integration of the MM4U framework and mobileMM4U framework allows to develop applications that dynamically create personalized multimedia content for both Desktop PCs and mobile devices.

The MM4U component framework is not only a generic architecture for applications in the domain of personalized multimedia content. With our modification of the hot-spot-driven development process for object-oriented frameworks towards the ProMoCF approach, we also provide for improving the development process of component frameworks. The experiences gained with applying and evaluating the ProMoCF approach for the development of the MM4U component framework are described in Section 8. The feedback retrieved from applying the outcome of the Pro-

MoCF approach, i. e., the MM4U component framework itself, for the development of concrete applications in the domain is presented in Section 10. In the following, we summarize the experiences gained in applying the ProMoCF approach. We also describe the feedback retrieved from developing the demonstrator applications that lead to an improvement and maturation of the ProMoCF approach.

With applying the ProMoCF approach for developing and evolving the MM4U component framework, we conducted an evaluation of our process model and development method for component frameworks. The experiences gained here are presented in the following.

- According to the ProMoCF approach, initially a set of hot-spots has been identified for the MM4U component framework and corresponding hot-spot-cards have been written. For identifying the hot-spots, our experience's gained in developing the prototypes of personalized multimedia applications as well as the analysis of related work in the area of generating personalized multimedia content were helpful. The initial hot-spot-cards have been re-worked by conducting several iterations of the ProMoCF approach's main cycle. In addition, the identified hot-spot-cards have been arranged into logical groups. With identifying and (re-)arranging the hot-spot-cards, five logical groups emerged for the MM4U component framework. Correspondingly, five group-hot-spot-cards were written. In the beginning of developing the MM4U component framework it was not clear how many group-hot-spot-cards would eventually be necessary for the framework. However, with conducting further iterations of the ProMoCF approach for developing and evolving the framework and applying the framework for the development of concrete applications, the number of group-hot-spot-cards finally meet to five. Each of the identified five group-hot-spot-cards addresses a particular task in the general multimedia personalization process as described in Section 5.1.
- With starting the development of the MM4U component framework, there had been some imagination of the flexibility requirements to the framework as well as the possible number of components needed for it. However, it was not clear, how many components would actually be necessary to realize these requirements and how to reasonably divide the flexibility requirements into distinct framework components. By identifying the five group-hot-spot-cards and developing a distinct framework component for each of these group-hot-spot-cards, we are sure that all necessary framework components have been identified. Another framework component cannot emerge, as all requirements to the framework (defined with the hot-spot-cards) have been successfully mapped to the identified group-hot-spot-cards and framework components, respectively. In addition, by employing the group-hot-spot-cards of the ProMoCF approach, we are sure that the emerged components are the right ones, i. e., that the flexibility requirements to the framework are reasonably divided into (the identified) components.
- For developing the MM4U component framework, we started with designing and implementing a first prototype of the framework based on the initially identified set of hot-spot-cards. This first framework prototype has then undergone constant review, re-design, and re-implementation iterations for a stepwise refinement and enhancement of the framework's components. These re-

design activities of the component framework were mainly triggered by the experiences gained with employing the framework for the development of the demonstrator applications in the field of multimedia personalization (see Section 10). Thus, the development of demonstrator applications are mainly responsible for ensuring the quality and evaluating the applicability of the MM4U component framework.

- Almost all component instances of the MM4U framework components are designed and implemented as traditional object-oriented frameworks. Only the Multimedia Composition component, which initial design was also an object-oriented framework, matured over time to a flexible and extensible toolbox for composing and assembling arbitrary personalized multimedia content. Thus, the ProMoCF approach allows not only for identifying the framework components but also for developing flexible component instances by means of object-oriented frameworks. As software components encapsulate their internal realization, the framework components shield the object-oriented frameworks from the outside world. Thus, the ProMoCF approach allows for evolving multiple object-oriented frameworks at the same time and supports the composition of these object-oriented frameworks by means of component technology into a component framework. By this, the ProMoCF approach allows for developing software frameworks of second order (see Section 4.1.2.1).
- As described in Section 8.9.2, we initially identified a hook method called doCompose(...) as hot-spot of the Multimedia Composition component. This hot-spot is required for providing complex and application-specific multimedia composition and personalization functionality. However, with evolving the Multimedia Composition component this hook method matured to be the constructor of the complex and sophisticated composition operators. Once the hook method is defined in the Multimedia Composition component, its signature is fixed and cannot be modified. Concrete applications would not be able to pass application-specific parameters to their complex operators to realize the required multimedia composition functionality. Thus, with refining the group-hot-spot-card of the Multimedia Composition component, we changed the initially identified hook method doCompose(...) to be the constructor of the concrete composition operators' classes. The constructor is more flexible, since its signature can be defined individually for each concrete complex or sophisticated composition operator. Thus, with applying the ProMoCF approach for developing the MM4U component framework, we learned that a constructor can be a hot-spot of a framework component.

The usage of the MM4U framework for the development of demonstrator applications is the second evaluation step of the ProMoCF approach. By this, the outcome of the ProMoCF approach, i. e., the quality of the MM4U component framework itself is evaluated. The experiences gained with applying the MM4U component framework are presented in the following. In addition, we describe the impact of the retrieved feedback on improving and maturing the ProMoCF approach.

- The development of demonstrator applications were used as valuation of the MM4U component framework. They gave us important feedback about the comprehensiveness and the applicability of the framework in order to improve

it in the subsequent iterations. The demonstrator applications pinpointed the weak points of the MM4U framework's architecture, i. e., they identified those parts of the framework design where it was too inflexible and additional hot-spots had been added. In addition, also those parts of the framework design with too much flexibility were identified such that the control flow of the application was unclear and ambiguous to the application developers. Consequently, additional frozen-spots had been added.

- During the development of the MM4U component framework and the first concrete demonstrator applications employing this framework, we observed that the granularity of hot-spots of about one method as defined in Pree's original hot-spot-driven design process is too imprecise (see Section 4.2.2.1). Comparing the flexibility requirements written on hot-spot-cards were difficult in order to identify groups of cards which have a high affinity in regard of solving a particular problem in the framework's domain. Due to the imprecise definition of hot-spot-cards, it was not clear how much and which functionality actually corresponds it. Thus, it was difficult to employ the MM4U component framework for the development of concrete applications, as this appliance requires clearly specified and unambiguous interfaces. This leads to the definition of the granularity of hot-spots in the ProMoCF approach to be exactly one (hook) method in the programming language (see Section 7.2). By this, we force the framework developers to clearly specify the functionality provided by a hot-spot-card and the corresponding service in the framework component.
- The initial version of the ProMoCF approach did not already come with the concept of group-hot-spot-cards. Once, we defined the granularity of hot-spot-cards to be exactly one hook method, we could clearly compare the flexibility requirements to the framework written on these cards. This allows us to identify groups of hot-spot-cards that describe flexibility requirements for a particular problem in the framework's domain. With evolving the MM4U framework and in particular with developing further applications as well as improving existing applications employing the framework, it emerged that the identified groups of hot-spot-cards are likely to determine the components of the MM4U framework. To make this knowledge explicit, we introduced the concept of group-hot-spots and group-hot-spot-cards, respectively. The group-hot-spot-cards are targeted at identifying the distinct components of a component framework and define the flexibility requirements to these components (see Section 7.2).

Although the ProMoCF approach has its limits as described in Section 7.3, it is a very valuable improvement towards a systematic software engineering support for developing component frameworks. This process model and its methodological support for identifying framework components and specifying flexibility requirements to these components allow computer scientists to design and evolve component frameworks more systematically than the *ad hoc* approach so far. Consequently, applying the ProMoCF approach provides for a better planning and implementation of future component frameworks.

12 Conclusion of this Work

Having presented the conceptual design and development of the MM4U component framework as well as the conceptual design and evaluation of the ProMoCF approach in the previous sections, we now give a summary of the results achieved in this work. As presented in Section 1.2, the goals of this work are twofold. Thus, the results achieved are twofold and presented in two sections. In Section 12.1, the results in regard of personalizing multimedia content with pursuing the MM4U component framework approach are described. In Section 12.2, a summary of the results in regard of improving the development process of component frameworks with the proposed ProMoCF approach is presented.

12.1 Personalization of Multimedia Content with the MM4U Approach

Although, there has been research in the field of multimedia personalization by different research groups for more than one and a half decade now, an integration approach like it is pursued with the MM4U component framework has not been proposed so far. The research results of our integration approach are described in the following sections along the structure of the goals of this work as presented in Section 1.2.

12.1.1 Framework Support for the Development of Personalized Multimedia Applications

The research conducted in this work aims at providing a component framework for supporting the development of personalized multimedia applications. With the MM4U component framework, we created an abstract and at the same time practical component architecture providing support for a generic development of such applications. Each component in this framework provides support for a different task in the general multimedia personalization process presented in Section 5.1 from media content and user profile access to final presentation delivery. With the MM4U component framework, we define and provide a concrete application architecture for personalized multimedia applications. The goal of the framework is to provide the basic tasks that occur with the generation of personalized presentations for application developers. Developers of personalized multimedia applications can rely on the predefined architecture of the MM4U framework and can immediately apply it for the development of their concrete application. Thus, they are not burden

with developing their own application architecture. In addition, the MM4U framework provides for reusing its component instances in different configurations for the development of personalized multimedia applications in different domains. These component instances can also be easily adapted to other specific requirements a concrete personalized multimedia application may have as well as newly developed component instances can be integrated into the MM4U framework.

12.1.2 Dynamic Generation of Multimedia Content in a Two-step Approach

As we argued in Section 3, the growing demand of creating personalized multimedia content can only result in providing support for a dynamic creation of multimedia content (see Section 5.1). As this can be achieved economically on larger scale only by a single source–multiple target generation of multimedia presentations, we pursue a two-step approach for the dynamic authoring of personalized multimedia content. In the first step, media elements are selected according to the user profile information and are arranged in time and space using an abstract multimedia content representation model. In the second step, the multimedia content in this internal representation model is transformed to the syntax and features of the concrete multimedia presentation formats.

For the design of the internal multimedia content representation model, we analyzed the different presentation formats that are supported by the end user devices today. The result is the design of an abstract multimedia content representation model that embeds the support of a wide range of today's presentation formats in regard of the central characteristics of multimedia modeling. These central characteristics are the definition of the multimedia presentation's spatial layout, temporal course, and interaction possibilities of the user with the presentation. The composition operators of our internal multimedia content representation model provide common and application-independent multimedia composition functionality. The representation model allows the integration of very specific multimedia composition and personalization functionality (see Section 6.3). Thus, the representation model of the MM4U framework allows for the composition of multimedia content in the most different application domains. In addition, the representation model is designed to be efficiently transformed by application independent transformation algorithms to the concrete syntax and features of the different presentation formats (see Section 6.4). This allows to provide support for a multi-channel generation of the personalized multimedia content in different presentation formats and for different targeted (mobile) end devices. This multi-channeling support is described next.

12.1.3 Support for Different End Devices and Standardized Multimedia Presentation Formats

The two-step approach of authoring personalized multimedia content, i. e., its composition in an internal multimedia content representation model and then the automatic transformation into different concrete presentation formats allows for both supporting different standardized presentation formats as well as different (mobile) end devices. Thus, we provide for a multi-channel generation of the personal-

ized multimedia content. Under standardized multimedia presentation formats, we summarize the formats which are provided by the W3C such as SMIL and SVG, the ISO standard LASeR, but also the industry-standard Flash. Supporting with the mobileMM4U framework also mobile profiles of these presentation formats such as SMIL BLP and Mobile SVG, we provide support not only for Desktop PCs but also for mobile end devices such as PDAs and cell phones.

The MM4U framework supports the authoring of multimedia content in standardized presentation formats as it is difficult in principal to develop an own multimedia player software for different end devices. This is especially important in the heterogeneous mobile world, where we find new devices every six month. The standardized multimedia presentation formats we find today can be employed as they are sufficient enough to provide for the essential characteristics of multimedia content. With employing standardized multimedia presentation formats for the multi-channel generation of personalized multimedia content, the application developers can rely on existing multimedia player software on the market. Thus, the application developers can focus on the multimedia content creation task and leave the creation of the final presentations to our multi-channel presentation generation chain. Consequently, the MM4U framework alleviates application developers from the burden of developing their own multimedia player software for different devices and lets them concentrate on the development of the actual application functionality. By this, we provide support for multimedia applications to overcome the last mile in creating and delivering appealing multimedia content to the end user's device. The explicit decision for presentation independence by a comprehensive internal multimedia content representation model and application independent transformation algorithms makes the framework both independent of any specific presentation format and prepares it for future formats to come.

12.1.4 Integration of Existing Solution Approaches and Systems on Different Levels

The MM4U framework is designed to integrate previous and existing research in the field. This design is based on the application scenarios described in Section 2, the extensive study and categorization of previous and ongoing related approaches in Section 3.2 and 3.3, and the derivation of the general process chain for multimedia content personalization in Section 5.1. The framework component interfaces explicitly allow to extend and exchange the framework's functionality. By this, the MM4U framework allows for integrating existing solutions and approaches for the specific tasks of the multimedia personalization process. Exchange of different multimedia personalization functionality is supported by configuring and exchanging the concrete instances of the MM4U framework components. For example, a new user profile store can be integrated by creating a new User Profile Accessor and Connectors component instance. The same applies for new media storage and retrieval solutions integrated by the Media Pool Accessor and Connectors component instance. In addition, arbitrary complex and application-specific multimedia composition and personalization functionality can be integrated with the Multimedia Composition component, e. g., a sophisticated composition operator exploiting a constraint solving approach, style sheets, templates, and others (see Section 3.3). As these approaches for multimedia content composition and personalization have

their individual advantages and disadvantages, we can exploit the advantages of each approach without necessarily having also to deal with their disadvantages. New or updated multimedia presentation formats can be easily embedded in our Presentation Format Generators component instance, as the framework's multimedia content representation model abstracts from the syntax and features of the concrete presentation formats. Finally, also the Multimedia Presentation component instance can be adapted to provide playback support for a specific multimedia presentation format. Since the framework is prepared at integrating current and future approaches and solutions in the field, it is of lasting value. New approaches and solutions can be integrated on different levels and thus the framework can be kept up to date to the most current technologies for the different tasks in multimedia content personalization.

12.1.5 Independence of the Actual Application Domain

The components of the MM4U framework are designed such that they provide support for arbitrary concrete applications that require multimedia personalization. Thus, they provide an application-independent support for the specific tasks in the general multimedia content personalization process. This is achieved for the Presentation Format Generators component by providing application-independent transformation algorithms to map the internal multimedia content representation model to the syntax and features of a specific presentation format. In addition, the component instance can be easily extended to support specific presentation formats a concrete application might need to deliver. The Multimedia Composition heavily depends on the actual application area. However, with the application-independent internal multimedia content representation model the component allows for creating arbitrary multimedia content in the domain of multimedia personalization. To provide enhanced support for the most different actual application areas in the domain of multimedia personalization, the internal representation model allows for extending by arbitrary application-specific composition functionality in form of complex and sophisticated composition operators. The User Profile Accessor and Connectors component and Media Pool Accessor and Connectors component define application-independent data models for integrating the most different user profile information and media data with associated meta data into the framework. These data models allow for specialization and refinement in regard of application-specific user profile information and media meta data. Thus, they provide support for the different concrete applications in the domain of multimedia personalization. Finally, the Multimedia Presentation component instance provides application-independent support for multimedia presentation playback by employing existing multimedia player software for standardized presentation formats. As this multimedia player software is typically application independent, also the Multimedia Presentation component instance can be employed for the most different concrete personalized multimedia applications. Providing application-independent support for the specific tasks in the general multimedia content personalization process makes the MM4U framework both resilient and robust in regard of the different personalization aspects that may appear with the different concrete personalized applications. By this, we provide for a stable and application-independent architecture in the domain of personalizing multimedia content.

12.2 Improvement of the Development Process of Component Frameworks with ProMoCF

Initially, the development process of component frameworks has been conducted *ad hoc*. As a consequence, the quality of component frameworks has been eventually dependent on the experience and skills of the framework developers only. The research presented in this thesis provides with the ProMoCF approach an improvement of this *ad hoc* process towards a more systematic development support for component frameworks. Following the structure of this work's goals in regard of improving the development process of component frameworks introduced in Section 1.2, the results are presented in the following sections.

12.2.1 Systematic Development of Component Frameworks

For a systematic development of component frameworks, we present in this work with ProMoCF a lightweight process model and development method for component frameworks. This process model for component frameworks bases on Pree's hot-spot-driven approach for developing object-oriented frameworks. The Pree process has been modified and enhanced in regard of defining the granularity of hot-spots and hot-spot-cards to be exactly one method in the programming language. In addition, activities for identifying the framework components as well as specifying, designing, implementing, and testing them have been defined. The ProMoCF approach has been evaluated in two ways. First, the approach has been applied for the development of the MM4U component framework. Second, the result of the ProMoCF approach, i. e., the MM4U component framework itself has been evaluated by employing it for the development of multiple applications in the domain of multimedia personalization. By employing the framework for the development of concrete personalized multimedia applications, the framework evolves and matures. In the ProMoCF approach, this application of the developed component framework is reflected and conducted by an explicit activity for applying and testing the framework.

12.2.2 Methods for Identifying and Specifying the Framework Components

The ProMoCF approach provides methodical support for identifying the framework components and specifying the flexibility requirements to these components by introducing the concept of group-hot-spots and group-hot-spot-cards (see Section 7.2). The concept of group-hot-spots is a consequent continuation and improvement of the original Pree process towards a development support for component frameworks. A group-hot-spot is a set of hot-spots that describe flexibility requirements for a common problem in the framework's domain. Thus, these hot-spots have a high affinity in regard of solving a particular problem that should be encapsulated in a distinct framework component. Consequently, the ProMoCF approach defines for each group-hot-spot-card a new framework component. The hot-spot-cards associated to the group-hot-spot-cards define the flexibility requirements to these components and provide for designing and implementing flexible component instances.

13 Open Issues and Future Work

Basing on the results of this work presented in the previous section, we consider open issues and possible future work. These are presented according to the summary of the results in two sections. In Section 13.1, open issues and possible future extensions in regard of the MM4U component framework are considered. In Section 13.2, open issues and aspects for improving the ProMoCF approach are discussed.

13.1 MM4U Component Framework

In regard of the MM4U component framework, we consider open issues and future work such as integrating a relevance-feedback component into the framework, extending the framework by further domain-specific approaches for multimedia personalization, providing framework support for authoring accessible multimedia content, and extending the framework in regard of deriving emergent semantics during the multimedia content authoring process. These issues and future work are presented in the following.

13.1.1 Extending the Framework by a Relevance Feedback Component

With relevance feedback, a mechanism is understood that provides the users the possibility to mark relevant and not relevant objects of the personalization result (cf. [BDLN04]). Thus, the users evaluate the results provided by the personalized multimedia application. The application then adapts its "performance" according to the user's feedback [DDV03]. This enables immediate response of the users' demands and preferences in the personalization results. A relevance feedback mechanism has been integrated, e. g., in our Sports4U application of a personalized multimedia sports news ticker presented in Section 10.2. However, this relevance feedback functionality is not targeted at providing an abstract, reusable relevance feedback component for the MM4U framework. Consequently, a possible future work is to extend the MM4U framework by a component for receiving and managing relevance feedback.

13.1.2 Extending The Framework by Further (Domain-specific) Approaches for Multimedia Personalization

In future work, further existing solution approaches for multimedia composition such as composition by constraints and rules or by document transformation (see Section 3.3) could be integrated into the MM4U framework. For example, a sophisticated multimedia composition operator could be developed that integrates the constrained-based Cuypers Multimedia Transformation Engine [OCG+00, GOH01]. Such a constraint-based approach would provide for a declarative authoring of personalized multimedia content. Also a template-based approach with selection operators could be applied for a future version of, e. g., our Sightseeing4U application presented in Section 10.1. This would allow domain experts for an easier exchange and manipulation of, e. g., the sightseeing presentations' content, layout, and design.

13.1.3 Extending the Framework by Authoring Accessible Multimedia Content

Another future extension of the MM4U framework could be to provide for authoring of accessible multimedia content. This means, that the MM4U framework provides for an explicit support for conveying the same information through multiple modality. The aim is to make same multimedia content accessible to different users with specific needs, e. g., for the visually or auditory impaired. Although the MM4U framework supports the creation of personalized multimedia content in different modalities, the framework is not targeted so far at providing a systematic support for authoring multimedia content in the different modalities and conveying the same information to different users with specific needs. For it, the Media Pool Accessor and Connectors component needs to provide explicit support for determining media elements for the presentations in different modalities and targeted users or user groups. These media elements of different modalities need to convey the same information, e. g., an audio element that provides an auditory description of an image element or a text element comprising a textual transcript of an audio element. According to the modality, the Multimedia Composition component needs to be able to assemble the selected media elements into coherent multimedia presentations that serve the requirements of the different modalities. Besides changing the modality, considering the disabilities of a targeted user or user group can also result in changing other characteristics of the presentation. For example, bigger icons should be used, e. g., for people with problems in their minute motor activity or with partly visually impaired, and one should avoid using the colors green and red for color-blind users.

13.1.4 Extending the Framework by Emergent Semantics

The MM4U component framework provides for the dynamic authoring of personalized multimedia content. Research on this matter has been conducted by different research groups for more than one and a half decade now. Today, we are pleased to see many scientific approaches and industrial solutions that provide such content to the user (see Sections 3.2 and 3.3). In recent years, however, it has becoming

increasingly clear that multimedia information does not have unique semantics but exhibits multiple semantics which depend on context and use [WMS05]. Although the systems and approaches we find today exploit the semantically rich meta data for their composition and assembly task, this highly valuable source of information is currently not considered any further. The actually created multimedia presentations carry none or only a small piece of the semantically rich meta data exploited for their creation. However, also with the combination of media elements into a coherent multimedia presentation new semantics can be derived from the used media elements' meta data. Creating personalized multimedia presentations for a specific user or user group even amplifies this effect. Thus, the MM4U framework could be extended in a future work to take the semantics into account that emerge during authoring personalized multimedia content. This extension of the MM4U framework would not only employ the semantics and meta data of the media elements for the composition and assembly task. It further would exploit the existing semantics and meta data for deriving higher-level semantics that emerge by combining media elements into new multimedia presentations and would bring this semantics to the end user. Consequently, the two challenging research fields of multimedia content personalization and emergent semantics for multimedia would be combined.

13.2 ProMoCF

In the previous section, open issues and possible future work in regard of the MM4U component framework are considered. In this section, we describe open issues and future work in regard of the lightweight process model and development method for component frameworks ProMoCF. This includes fostering the ProMoCF approach by conducting further evaluation studies and improving the ProMoCF approach by providing support for further aspects of scale and granularity of software components [SGM02].

13.2.1 Fostering the ProMoCF Approach by Further Evaluation Studies

Besides evaluating the ProMoCF approach for the development of the MM4U component framework and the application of this component framework for the development of concrete personalized multimedia applications, further evaluation studies are needed to foster the ProMoCF approach. This means that the ProMoCF approach needs to be applied for the development of component frameworks in other domains. For example, the ProMoCF approach could be applied for the development of component frameworks in the domain of business software such as billing systems for telecommunication providers, financial software, or other possibly more specific domains.

However, a further evaluation of the ProMoCF approach could also be conducted in the area of multimedia player software. For the MM4U component framework, we target with the Multimedia Presentation component at exploiting existing player software for rendering multimedia presentations in standardized formats such SMIL, SVG, and the industry-standard Flash. We aim at exploiting existing player software due to the high effort involved with the development and mainte-

nance of an own multimedia player, especially when developing such a player for different mobile end devices. Consequently, it is a big challenge for companies aiming at developing a multimedia player for the different operating systems and devices. What is needed here, is a flexible software architecture that allows for a fast and robust development of multimedia players for different end devices and platforms. This multimedia player architecture could be defined by a component framework developed with the ProMoCF approach. However, such a flexible multimedia player framework should not be targeted at presenting complex presentation descriptions such that they are possible with SMIL and SVG. Parsing and processing presentations in these formats is a very complex task which adds additional effort to the multimedia player. However, a very slim and simple structured but nevertheless very powerful document model should be employed (in contrast to the approach by Honkala et al. [HCV04] targeting at employing SMIL and XForms [XFo06] for a platform independent multimedia player). The binary Flash format has proved to be very useful as it is small, fast, and the definition of the presentations' temporal and spatial structure as well as interaction possibilities is straight forward by employing a global timeline and an absolute spatial positioning model. As a consequence, the proposed multimedia player software could be designed to natively playback the FML developed in this work. As a straightforward XML-representation of the binary Flash format, the FML has some major advantages for developing a multimedia player compared to SMIL or SVG. It is very simple and it will be easy to develop a multimedia player for it. Such a player is not burden with the complex tasks of parsing nested document structures, representing different local timelines, and the like. As the FML uses a global timeline and absolute positioning model, the playback of the multimedia presentation can be implemented as a merely sequential processing of the FML. In principle, a format such as the FML would enable the multimedia player to present the content while it is streamed over the network.

13.2.2 Extending the ProMoCF Approach by Further Framework Aspects

In the work by Szyperski et al. [SGM02], different aspects of scale and granularity for software components are introduced. As presented in Section 7, the ProMoCF approach considers these aspects of scale and granularity in regard of components being units of analysis. In a future improvement of the ProMoCF approach, the process model and development method could be extended in regard of considering software components as being units of dispute, locality, and loading [SGM02, 140ff.].

Components as Units of Dispute An interesting challenge with component software and thus with component frameworks is the propagation of errors [ALRL04] across component boundaries. Normally, software components are aimed to handle errors by themselves. It cannot be expected that other components have the necessary inside knowledge to do so. However, some errors cannot be handled locally within the component but need to be propagated to the calling component. Such propagated errors are observed by the calling component as service failures [ALRL04]. For example, consider a component retrieving data from a remote database. When executing a service of this component, it might be confronted with technical problems such as

a network failure or database breakdown. Consequently, the component will not be able to provide for a successful delivery of the requested service. An error is raised within the component that is propagated to the calling component as service failure. This service failure needs to be declared as part of the component contract. The calling component is expected to handle the service failure in an appropriate manner or again to propagate it properly. So far, the ProMoCF approach does not explicitly provide for identifying and specifying such errors and service failures. Consequently, the exception concept defined in the MM4U framework (see Section 8.13.2) bases more on experience than on a systematic approach. Thus, a future version of the ProMoCF approach might be extended by providing explicit support for identifying service failures that shall be declared in the component contracts and by this provide support for a more systematic handling of errors.

Components as Units of Locality For component software, it is important to have a good deployment plan of the components. Such a deployment plan determines where the components are actually deployed to and installed on, e. g., a server machine or a client device. This is important, since the deployment changes the communication latency between the components. Thus, it can influence the performance of a component framework significantly. As a consequence, the aspect of considering components as units of locality needs to be taken into account when determining a deployment plan for a component framework. This means that for the framework components it must be determined whether they are executed on the same machine and processor, are communicating via a local area network, or are even deployed on systems connected over a wide area network. For the MM4U component framework, the aspect of locality is important in regard of the Multimedia Composition component and the Media Pool Accessor and Connectors component. As there is a high communication load between these two components, they should be strongly coupled, i. e., preferably executed on the same machine or within a local area network. A future extension of the ProMoCF approach could provide for considering the deployment of the framework components and thus take the aspect of locality into account.

Components as Units of Loading The aspect of components as units of loading considers the problems that can occur when loading a new version of a component into an already running software system. Thus, considering components as units of loading is closely linked to the aspect of components as units of maintenance. With units of maintenance, the challenges are meant that have to be considered when maintaining and updating a software system. When maintaining a component framework, i. e., when evolving the framework, new versions of the framework components emerge. These new versions of the framework components can possibly not match some framework components of an older version. When updating an application with the evolved framework and framework components it may come to version mismatches (fragile base class problem [SGM02, WS96]). For example, an application could for any reason need to use an older version of a framework component. One possibility to alleviate this version mismatch problem is to provide for installing and loading multiple versions of the same framework component into the application (see side-by-side-loading in [SGM02, 148ff.]). However, this does

not work in all cases, e. g., if the framework component manages external resources such as information stored in a database. For example, with the MM4U component framework it is possible to have two or more versions of, e. g., the Media Pool Accessor and Connectors component and Presentation Format Generators component installed and loaded into the same application. However, loading of different variants of the User Profile Accessor and Connectors component might raise version conflicts as the component provides for storing concrete user profiles, e. g., on a file system or in a database. In a future improvement of the ProMoCF approach, it could be desirable to have version support when evolving the framework. This would allow for dealing with at least some incompatibility problems due to version mismatches in order to keep the applications up to date.

Appendix

A MM4U Framework Cookbook

When developing a new personalized multimedia application with the MM4U framework, many architectural design decisions are already conducted and predefined by the framework. For using and adapting a framework for the development of a concrete application, e. g., a corresponding framework cookbook (see Section 4.2.1) can be provided. Such a framework cookbook supports and guides the application developers with understanding the framework. In addition, it supports the users in adapting the framework to the requirements of a concrete personalized multimedia application. For it, the cookbook provides recipes, i. e., informal descriptions for conducting specific tasks in using and adapting the framework.

For using the MM4U component framework for developing concrete personalized multimedia applications, the framework component instances are reused and adapted to the specific requirements of the application. From our experiences with developing the applications presented in Section 10, we can say that it takes one day up to a week to gaining the necessary knowledge to apply and adapt the MM4U framework.[1] In the following, an excerpt of the MM4U framework cookbook is presented. As a full description of the MM4U framework cookbook is beyond the scope of this thesis, this excerpt focuses on recipes covering central tasks of (re-)using and adapting the framework. These are the reuse of existing component instances in different configurations, the development of application-specific multimedia composition and personalization functionality, as well as the development and actual execution of a concrete personalized multimedia application. We show these tasks along the development of a simple slideshow application introduced in Section A.1. In Section A.2 to A.4, we present three recipes for conducting the central usage and adaptation tasks and apply them for the development of our slideshow application.

A.1 Description and Requirements of the Slideshow Application

The slideshow application aims at presenting a personalized sequence of pictures taken from a photo album according to the user's preferences. These preferences are the selection criteria for the photos and comprise exposure, similarity, and sharpness of the photos, the playback duration per slide, the use of a background music, and the targeted output format for the presentation. Thus, the slideshow application constitutes a simplified variant of the Pictures4U application presented in Section 10.5. Like the Pictures4U application, the envisioned simple slideshow ap-

[1] Statements of students who used and adapted the framework.

plication shall provide its personalized multimedia content to the users over the Internet for consumption. For developing the slideshow application different tasks have to be conducted. Each task raises a specific question how to apply and adapt the MM4U component framework.

First, the existing component instances of the MM4U framework shall be reused where possible. For the slideshow application, we aim at employing the instances of the User Profile Accessor and Connectors, Media Pool Accessor and Connectors, Presentation Format Generators, as well as Multimedia Presentation component the MM4U framework already provides. These component instances need to be configured according to the requirements of the slideshow application. Thus, the question raises how to configure existing MM4U component instances in order to reuse them in a specific personalized multimedia application?

Second, a new instance of the Multimedia Composition component needs to be designed and implemented providing the required multimedia composition and personalization functionality of the slideshow application. This is realized by developing a new sophisticated composition operator for the MM4U framework. Thus, the question raises how such a composition operator can be designed and implemented?

Finally, in order to employ the MM4U framework for the slideshow application it needs to implement an instance of the MM4U component. The development of this instance includes creating an appropriate environment for executing the personalized application. As the slideshow application shall be available via the Internet to its users, we aim at providing it as Java servlet. Consequently, the last questions are how to develop an instance of the MM4U component and how to apply it for the development and execution within a Java servlet for the Internet?

The questions raised by the framework usage and adaption tasks introduced above are answered by the framework cookbook recipes presented in the following sections. Based on [FPR02, Pre95b], each recipe is roughly structured into three parts. First, the purpose of the recipe is introduced. Then, a procedure is presented how to apply the recipe. For illustrating the employment of the recipe, the development of our slideshow application as well as some source code example(s) are presented.

A.2 Recipe of How to Configure the Existing Instances of the MM4U Framework Components

Purpose The purpose of this cookbook recipe is to present how to (re-)use specific component instances of the MM4U framework in different concrete personalized multimedia applications. These component instances shall be adapted to the requirements of the concrete application by applying them in different configurations only. Thus, this adaptation of the component instances and their behavior is conducted without source code modification. Such a configuration of component instances can be typically applied for the User Profile Accessor and Connectors, Media Pool Accessor and Connectors, Presentation Format Generators, and Multimedia Presentation component of the MM4U framework.

Procedure The component instances are configured by passing appropriate parameters to the corresponding getFactory(...) method. This method internally configures the component and creates a component instance with all required objects and that provides a specific component behavior as specified by the parameters' values.

For the Presentation Format Generators component instance, this is achieved by passing the targeted presentation format as integer value to the getFactory(*<targetFormat>*) method. For it, the GeneratorToolkit class implementing this method provides different constants defining the values for the supported presentation formats. For example, passing the constant SMIL2_0 to the getFactory(*<targetFormat>*) method creates an instance of the Presentation Format Generators component that provides for transforming multimedia content in the internal representation model to the syntax and features of the SMIL 2.0 presentation format. Other constants provided by the GeneratorToolkit class are, e. g., SVG1_2 for SVG 1.2, FLASH for Adobe's Flash format, and REALPLAYER_SMIL2_0 for creating a player specific generator for RealPlayer's extension of the SMIL format in regard of formatted text elements.

For the User Profile Accessor and Connectors, Media Pool Accessor and Connectors, and Multimedia Presentation component instances of the MM4U framework providing a configuration by a getFactory(...) method, the configuration parameter is not provided as an integer value but by employing the locator concept introduced in Section 8.7.3. Here, the parameters required for configuring a specific component instance are passed via a locator object to the getFactory(...) method. As described in Section 8.7.3, the parameters that are to be passed to the getFactory(...) method in order to obtain a specific component behavior depend on the concrete functionality the application developers require. For example, the parameters of a locator object for creating a Media Pool Accessor and Connectors component instance to a relational database are much different to a component instance accessing media elements from a web server via the URIMediaElementsConnector.

When applying the MM4UApplicationRunner class (see Section 8.12.2) for executing a personalized multimedia application, the getFactory(...)-calls for creating the component instances are hided from the application developers for reasons of convenience. The application of the MM4UApplicationRunner class is described in the framework cookbook recipe presented in Section A.4.

Examples Reusing the Presentation Format Generators component instance for the generation of a specific presentation format is very simple. An example source code for creating such a component instance for the Flash format is like follows:

```
IGenerator flashGenerator =
  GeneratorToolkit.getFactory( GeneratorToolkit.FLASH );
```

For creating an instance of Media Pool Accessor and Connectors component providing access to media elements stored on a web server in the Internet, the URIMediaElementsConnector can be applied. A corresponding implementation of a locator object and creation of the Media Pool Accessor and Connectors component instance is shown in the following:

```
IMediaElementsConnectorLocator mediaConnectorLocator =
  new URIMediaElementsConnectorLocator(
    "http://www-is.informatik.uni-oldenburg.de/~mm4u/Media/",
    "offis-mediaserver.index" );
```

```
5 IMediaElementsAccessor mediaConnector =
    MediaElementsAccessorToolkit.getFactory( mediaConnectorLocator );
```

A.3 Recipe of How to Develop a Composition Operator

Purpose The multimedia composition and personalization functionality is heavily dependent of the concrete application's domain (see Section 8.14). Thus, typically a new instance of the Multimedia Composition component needs to be developed. This recipe describes how application developers can create their own multimedia composition functionality. For it, they can reuse existing multimedia composition functionality the MM4U framework provides as well as employing the functionality available from other concrete personalized multimedia applications.

Procedure Developing a new instance of the Multimedia Composition component means to implement the IComplexOperator interface or extend its abstract implementation the AbstractComplexOperator class. This interface and abstract class are described in Section 8.9.3. For developing a new multimedia composition operator a corresponding class is created, which is implementing the IComplexOperator interface or extending the AbstractComplexOperator class. One needs to decide, which and what kind of media elements are needed for the operator. Then, the temporal course of the multimedia content that shall be created by the operator is defined. Here, the selected media elements are assembled by applying appropriate composition operators. In addition, the layout of the content is specified in regard of, e. g., the arrangement of visual media elements in space as well as the acoustic layout of audio elements and video elements. For it, appropriate projectors are defined and added to the composition operators and media elements. As the example below will show, this is not necessarily a sequential procedure but can also be conducted in different smaller steps. The order in which the selected media elements, composition operators, and projectors are assembled is dependent on the actual multimedia content to be created as well as application developers' preferences and styles. One possible way to assemble the personalized multimedia content is to proceed from the leaves to the root of the presentation. First, the leaves, i. e., the media elements of the object-oriented multimedia content tree are created. These are assembled to larger units by means of basic, complex, and sophisticated composition operators. The resulting presentation parts are then combined unless the desired personalized multimedia content has been created.

Example From the description of our slideshow application in Section A.1, we derive that the Multimedia Composition component instance for this application needs to present a sequence of pictures together with an optional background music. The duration of presenting each picture or slide needs to be variable according the user's preferences. When the presentation of the pictures finishes, also the playback of the audio element shall stop. The result is the sophisticated composition operator SlideshowWithBackgroundMusic presented in Listing A.1. For reasons of convenience we base on the AbstractComplexOperator class to implement this operator rather than

employing the IComplexOperator interface. The constructor of this operator takes as parameter all information required to fulfill the desired multimedia composition functionality (see line 3 to 6). For assembling the slideshow's content, first the sequence of the pictures is created (line 10 to 20). For each photo in the photoList a TemporalSelector is created with the user's preferred playback duration. This selector is applied on the photos and an appropriate SpatialProjector is added. The presentation slides are then added to a Sequential operator. Having created the single pages of the slideshow, a logo of the MM4U framework is added to the presentation (line 22 to 29). This logo is variable and is passed as parameter to the SlideshowWithBackgroundMusic operator. The slideshow is placed directly below the logo, depending on its height (line 31 to 37). Then, the root operator of the slideshow is created (line 40). Finally, if the user wishes a background music for the slide presentation, a fixed audio element called vienna4u_audio is added (line 42 to 51). Here, we employ the already provided complex composition operator BackgroundMusic of the MM4U framework. Whether the users wish background music or not, the creation of the multimedia content is finished by calling the setRootOperator(...) method provided by the AbstractComplexOperator class (line 48 and 50). This enables the Presentation Format Generators component to retrieve the multimedia content assembled by the SlideshowWithBackgroundMusic operator via the getRootOperator() method of the IComplexOperator interface (see Section 8.9.3).

```
public class SlideshowWithBackgroundMusic extends AbstractComplexOperator {

  public SlideshowWithBackgroundMusic(IMediaList photoList,
      int durationPerSlide, IImage logo, boolean usebackgroundMusic,
5     IUserProfile userProfile, IMediaElementsAccessor mediumConnector )
      throws MM4UMediumElementNotFoundException {
    // Determine the number slides
    int numberOfSlides = photoList.size();

10  // Create a sequence of the pictures
    ISequential slideSequence = new Sequential();
      for (int slideCounter = 0; slideCounter<numberOfSlides; slideCounter++) {
        IImage photo = (IImage)photoList.elementAt( slideCounter );
        ITemporalSelector temporalSelectorPhoto =
15        new TemporalSelector( photo, 0, durationPerSlide );
        ISpatialProjector spatialProjectorPhoto =
          new SpatialProjector( 0, 0, photo.getWidth(), photo.getHeight(), 0 );
        temporalSelectorPhoto.addProjector( spatialProjectorPhoto );
        slideSequence.addVariable( temporalSelectorPhoto );
20    }

    // Add a logo to the slideshow
    int logoWidth  = logo.getWidth();
    int logoHeight = logo.getHeight();
25  ISpatialProjector spatialProjectorLogo =
      new SpatialProjector( 0, 0, logoWidth, logoHeight, 0 );
    logo.addProjector( spatialProjectorLogo );
    ITemporalSelector temporalSelectorLogo =
      new TemporalSelector( logo, 0, numberOfSlides * durationPerSlide );
30
    // Determine the with and height of the slideshow area for the projector
    IPresentationFragmentBoundaries boundaries =
      GeneratorToolkit.getFragmentBoundaries( slideSequence );
    // Add the slides presentation directly below the logo
35  SpatialProjector contentProjector = new SpatialProjector(
```

```
        0, logoHeight, boundaries.getWidth(), boundaries.getHeight(), 0);
      slideSequence.addProjector(contentProjector);

      // Create the slideshow presentation's root operator
      IParallel slideshow = new Parallel( temporalSelectorLogo, slideSequence );

      // Add a pre-determined background music to the presentation (optional)
      if ( usebackgroundMusic ) {
        IAudio backgroundMusic = (IAudio)mediumConnector.getMediumElement(
          "vienna4u_audio");
        IOperator rootOperator = new BackgroundMusic( slideshow, backgroundMusic,
          0, numberOfSlides * durationPerSlide );
        this.setRootOperator( rootOperator );
      } else {
        this.setRootOperator( slideshow );
      }
    }
  }
```

Listing A.1: Implementation of the sophisticated composition operator SlideshowWithBackgroundMusic

A.4 Recipe of How to Develop and Execute a Personalized Multimedia Application

Purpose This recipe of the MM4U framework cookbook provides application developers support for developing and executing their personalized multimedia application. It allows for a fast design and implementation of a concrete personalized application, i. e., creating an instance of the MM4U component. In addition, it is described how to execute this personalized multimedia application as a Java servlet.

Procedure Creating an instance of the MM4U component means to implement its services as specified in the provided interface IPersonalizedMultimediaApplication in Section 8.12.1. With implementing the provided interface's services, the application developers bring in their application-specific functionality into the MM4U framework and can adapt the framework according to the application's requirements.

For executing a personalized multimedia application implementing the IPersonalizedMultimediaApplication interface, the MM4UApplicationRunner class provided by the MM4U framework can be applied (see Section 8.12.2). This application runner is employed with the concrete personalized multimedia application as one of its parameter. Further parameters of the application runner are the locator objects for the User Profile Accessor and Connectors component and Media Pool Accessor and Connectors component, the targeted presentation format (if not specified by or exploited from the user profile information), the locator object of the Multimedia Presentation component, the profile ID of the current user, as well as any additional parameters that shall be passed. When executing the application, the MM4UApplicationRunner class calls the services of the IPersonalizedMultimediaApplication interface as depicted by the sequence diagram in Figure 8.17 on page 201. Here, the application-specific

code implemented by the application developers is called by the MM4U framework according to the call-back-principle (see Section 4.1.1).

For embedding the developed personalized multimedia application in a Java servlet, the MM4U framework provides the ExtendedHttpServlet class. This class is a slight extension of the HttpServlet class provided by Java. It extends it in regard of reading the servlet's header and parameters and integrating these parameters into the data structures provided by the MM4U framework. Within an application-specific subclass of the ExtendedHttpServlet, the execution of the concrete personalized multimedia application by means of the MM4UApplicationRunner is called. Thus, within this subclass also the application runner's parameters as numerated above are specified and initialized.

Example The concrete implementation of the IPersonalizedMultimediaApplication interface for the envisioned slideshow application is shown in Listing A.2. The listing shows the SampleApplication class and the application-specific implementations for the personalized slideshow. The implementation of the services existUserProfile(...), getUserProfile(...), deleteUserProfile(...), and setUserProfile(...) is straight forward and directly applies the passed user profile connector (line 3 to 12, 31 to 34, and 66 to 70). However, for creating new user profiles with the createUserProfile(...) service not an empty profile is created by passing the call to the provided profile connector (line 14 to 29). Instead, a template profile with the ID template_user_profile is loaded. This user profile template comes with default values for the personalization aspects relevant for the slideshow application. A very important service of the slideshow application is updateUserProfileInformation(...) (line 36 to 66). It provides for updating the user profile information with the additionalProperties, i. e., the parameter values retrieved from the slideshow servlet. If these parameters are provided by the servlet's web page, the user profile information is updated. The values for the targeted presentation format, use of background music, and presentation duration for the slides is determined and stored into the user profile.

The implementation of the doCompose(...) service provides for integrating the application-specific multimedia composition and personalization functionality (line 73 to 99). First, the user's presentation preferences are extracted from the corresponding profile. Then, the photo album to be presented is determined. This is extracted from the additionalProperties provided by the web page and slideshow servlet, respectively. In the next step, a query object for retrieving the photos that match the user profile information is created within the createQueryObject(...) method (line 101 to 138). Here, the query object's parameters are specified such as if the users like to include the photos' exposure, sharpness, and similarity as parameters into the selection process. Then, the query object is executed and a list of best matching photos is determined (line 89 to 90). The MM4U logo is retrieved from the media store (line 93), before the sophisticated composition operator SlideshowWithBackgroundMusic presented in Section A.4 is employed for actually composing the content (line 97 to 98).

With the implementation of the doTransform(...) service, the instance of the Presentation Format Generators component is called (line 140 to 147). In addition, it specifies the title of the slideshow presentation. Finally, the doPresent(...) service employs the multimedia player software of the Multimedia Presentation component (line 149 to 152).

```
public class SampleApplication implements IPersonalizedMultimediaApplication {

  public boolean existUserProfile(IUserProfileAccessor profileConnector,
      String profileID) {
    return profileConnector.existUserProfile(profileID);
  }

  public IUserProfile getUserProfile(IUserProfileAccessor profileConnector,
      String profileID) throws MM4UUserProfileNotFoundException,
      MM4UInvalidUserProfileException {
    return profileConnector.getUserProfile(profileID);
  }

  public IUserProfile createUserProfile(IUserProfileAccessor profileConnector,
      String profileID) throws MM4UCannotCreateUserProfileException {
    // Do not apply the createUserProfile(...) service for creating a new user
    // profile but employ a pre-defined template user profile instead.
    IUserProfile userProfile = null;
    try {
      profileConnector.getUserProfile("template_user_profile");
    } catch ( MM4UUserProfileNotFoundException exception ) {
       throw new MM4UCannotCreateUserProfileException(this,
         "createUserProfile()", "Default template user profile not found.");
    } catch ( MM4UInvalidUserProfileException exception ) {
       throw new MM4UCannotCreateUserProfileException(this,
        "createUserProfile()", "Requested user profile is invalid." );
    }
    return userProfile;
  }

  public void deleteUserProfile(IUserProfileAccessor profileConnector,
      String profileID) throws MM4UCannotDeleteUserProfileException {
    profileConnector.deleteUserProfile(profileID);
  }

  public IUserProfile updateUserProfileInformation(IUserProfile userProfile,
      IPropertyList additionalProperties)
      throws MM4UUserProfilesConnectorException {
    // Update user profile if presentation format is provided via the servlet
    int userSelectedPresentationFormat = additionalProperties.getIntegerValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "presentationFormat" );
    if (userSelectedPresentationFormat != Constants.UNDEFINED_INTEGER) {
      IUserProfileNode profileNode =
        userProfile.getNodeByPath("user.preferences.presentation_format");
      profileNode.setValue(userSelectedPresentationFormat);
    }
    // Update use of background music
    boolean useBackgroundMusic = additionalProperties.getBooleanValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "useBackgroundMusic");
    if ( useBackgroundMusic ) {
      IUserProfileNode profileNode =
        userProfile.getNodeByPath("user.preferences.backgroundMusic");
      profileNode.setValue(useBackgroundMusic);
    }
    // Update playback duration per slide
    int durationPerSlide = additionalProperties.getIntegerValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "durationPerSlide");
    if ( durationPerSlide!=Constants.UNDEFINED_INTEGER && durationPerSlide>0 ) {
      IUserProfileNode profileNode =
        userProfile.getNodeByPath("user.preferences.durationPerSlide");
```

```
      profileNode.setValue(durationPerSlide);
    }
    // Update further personalization aspects such as the maximum number of
    // photos to be shown...
    return userProfile;
  }

  public void setUserProfile(IUserProfileAccessor profileConnector,
      IUserProfile userProfile) throws MM4UCannotStoreUserProfileException {
    profileConnector.setUserProfile(userProfile);
  }

  public IVariable doCompose(IUserProfile userProfile,
      IPropertyList additionalProperties,
      IMediaElementsAccessor mediumConnector)
      throws MM4UMediumElementNotFoundException {
    // Read user preferences from the user profile
    int durationPerSlide =
      userProfile.getIntegerValue( "preferences.slideshow.durationPerSlide" );
    boolean backgroundMusicEnabled =
      userProfile.getBooleanValue( "preferences.slideshow.useBackgroundMusic" );

    // Determine the album for the personalized slideshows from servlet
    String photoAlbum  = additionalProperties.getStringValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "albumName" );
    // Create a query object to retrieve photos and the execute query
    IQueryObject queryParameters = this.createQueryObject(
      photoAlbum, additionalProperties );
    IMediaList imageList = mediumConnector.getMediaElements(
      queryParameters, userProfile);

    // Get the MM4U logo
    IImage logo = (IImage)mediumConnector.getMediumElement("mm4u_logo");

    // Apply sophisticated multimedia composition operator to create the
    // personalized slideshow
    return new SlideshowWithBackgroundMusic(imageList, durationPerSlide, logo,
      backgroundMusicEnabled, userProfile, mediumConnector);
  }

  private IQueryObject createQueryObject( String photoAlbum,
      IPropertyList additionalProperties ) {
    // Create query object and specify parameters
    IQueryObject queryParameters = new QueryObject();
    String albumPath = additionalProperties.getStringValue( "albumPath" );
    queryParameters.put(QueryObject.ATTRIBUTE_URI, albumPath + photoAlbum );
    queryParameters.put(QueryObject.ATTRIBUTE_FILTER_LIST,
      MediaTranscoderAndTransscaler.getReadExtensions());
    queryParameters.put(QueryObject.ATTRIBUTE_FIT, IQueryObject.FIT_MEET);
    queryParameters.put(QueryObject.ATTRIBUTE_WIDTH, 800 );
    queryParameters.put(QueryObject.ATTRIBUTE_HEIGHT, 600 );

    // Employ exposure, similarity, and sharpness for photo selection
    if (additionalProperties.getBooleanValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "useExposure")) {
        queryParameters.put(PhotoSelection.ATTRIBUTE_MDA_EXPOSURE, 0.75);
    } else {
      queryParameters.put(PhotoSelection.ATTRIBUTE_MDA_EXPOSURE,
        Constants.UNDEFINED_FLOAT);
    }
```

```
    if (additionalProperties.getBooleanValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "useSimilarity")) {
        queryParameters.put(PhotoSelection.ATTRIBUTE_MDA_SIMILARITY, 0.2);
    } else {
      queryParameters.put(PhotoSelection.ATTRIBUTE_MDA_SIMILARITY,
        Constants.UNDEFINED_FLOAT);
    }
    if (additionalProperties.getBooleanValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "useSharpness")) {
        queryParameters.put(PhotoSelection.ATTRIBUTE_MDA_SHARPNESS, 0.2);
    } else {
      queryParameters.put(PhotoSelection.ATTRIBUTE_MDA_SHARPNESS,
        Constants.UNDEFINED_FLOAT);
    }
    // Add further querying parameters such as determining a maximum number of
    // photos to be selected...
    return queryParameters;
  }

  public IMultimediaPresentation doTransform(IGenerator generator,
      IVariable rootVariable, int width, int height, IUserProfile userProfile)
      throws MM4UGeneratorException {
    String title = "Slideshow with background music " +
      Constants.COPYRIGHT_STATEMENT;
    return generator.doTransform(rootVariable, title, width, height,
      userProfile);
  }

  public void doPresent(IMultimediaPlayer player,
      IMultimediaPresentation presentation) throws MM4UPlayerException {
    player.play(presentation);
  }
}
```

Listing A.2: Implementation of the IPersonalizedMultimediaApplication interface for the slideshow application

As described in the procedure paragraph of this section, the implementation of the IPersonalizedMultimediaApplication interface, i. e., the SampleApplication class is passed as parameter to the MM4UApplicationRunner class to actually execute the application. This embedding of the SampleApplication class in the slideshow servlet and ExtendedHttpServlet class is shown in Listing A.3. The concrete servlet class for the slideshow application is called SampleServlet. It communicates with the clients via the doGet(...) and doPost(...) methods (the second method only refers to the first one). Within the doGet(...) method, first the parameters provided by the servlet are extracted (line 7 to 8). Then, a parameter is added determining the root path for the photo albums (line 10). In the next step, the locator objects for configuring the User Profile Accessor and Connectors, Media Pool Accessor and Connectors, and Multimedia Presentation component are specified (line 12 to 18). As the slideshow application shall be realized as Java servlet, the HttpServletPlayer of the Multimedia Presentation component instance is employed for presenting the multimedia content in the final presentation format via an appropriate multimedia player software on the user's end device. To execute the slideshow application, an instance of the SampleApplication is created (line 27), before the slideshow application is actually executed by applying the MM4UApplicationRunner class (line 29 to 39).

```
public class SampleServlet extends ExtendedHttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws IOException {
    // Retrieve parameters from servlet
    // (a feature of the ExtendedHttpServlet class)
    IPropertyList additionalProperties =
      this.getServletParametersAndHeader(request);
    // Add the photo album path to the additional properties
    additionalProperties.add("albumPath", "file:///D:/Pictures/");

    // Create locator objects
    IUserProfileConnectorLocator profileConnectorLocator =
      new URIUserProfileConnectorLocator("file:///D:/Profiles/");
    IMediaElementsConnectorLocator mediaConnectorLocator =
      new URIMediaElementsConnectorLocator(
        "http://www-is.informatik.uni-oldenburg.de/~mm4u/Media/",
        "offis-mediaserver.index");
    IMultimediaPlayerLocator playerLocator =
      new HttpServletPlayerLocator(response);

    // Determine user profile ID from parameters provided by the servlet
    String userProfileID = additionalProperties.getStringValue(
      ExtendedHttpServlet.SERVLET_PARAMETER + "profileID");

    // Create instance of the sample slideshow application
    SampleApplication slideshowApplication = new SampleApplication();

    // Run the slidshow application by means of the MM4UApplicationRunner
    try {
      MM4UApplicationRunner.runPersonalizedMultimediaApplication(
        slideshowApplication, profileConnectorLocator, mediaConnectorLocator,
        MM4UApplicationRunner.USE_PRESENTATION_FORMAT_FROM_USER_PROFILE,
        playerLocator, userProfileID, additionalProperties);
    } catch (Exception exception) {
      String errorMsg = "Error occurred executing the slideshow application: " +
        exception.getMessage();
      this.writeOutputError(response, errorMsg, exception);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response)
      throws IOException {
    this.doGet(request, response);
  }
}
```

Listing A.3: Implementation of the slideshow servlet and execution of the personalized multimedia application

B Specification of the Internal Multimedia Content Representation Model in UML

Complete specification of the internal multimedia content representation model in UML. For a detailed description of this representation model see Section 6.3. In addition, also a DTD of the basic multimedia composition elements of the internal representation model can be found in Appendix C.

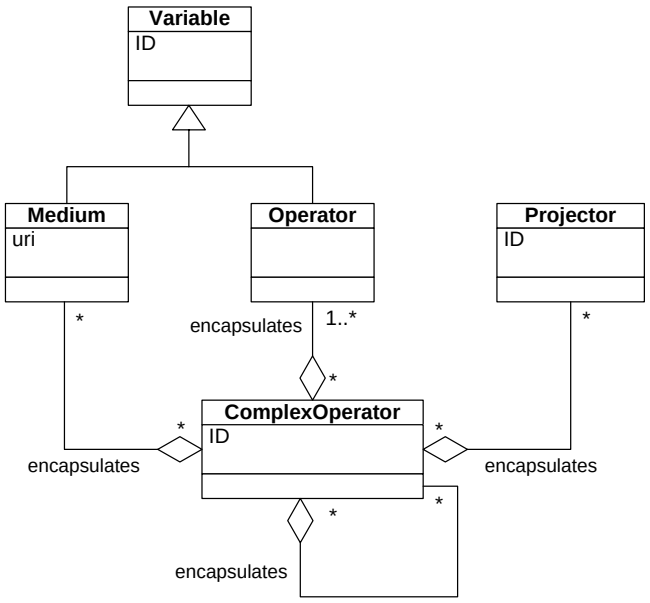


Figure B.1: The core concept of the internal multimedia content representation model

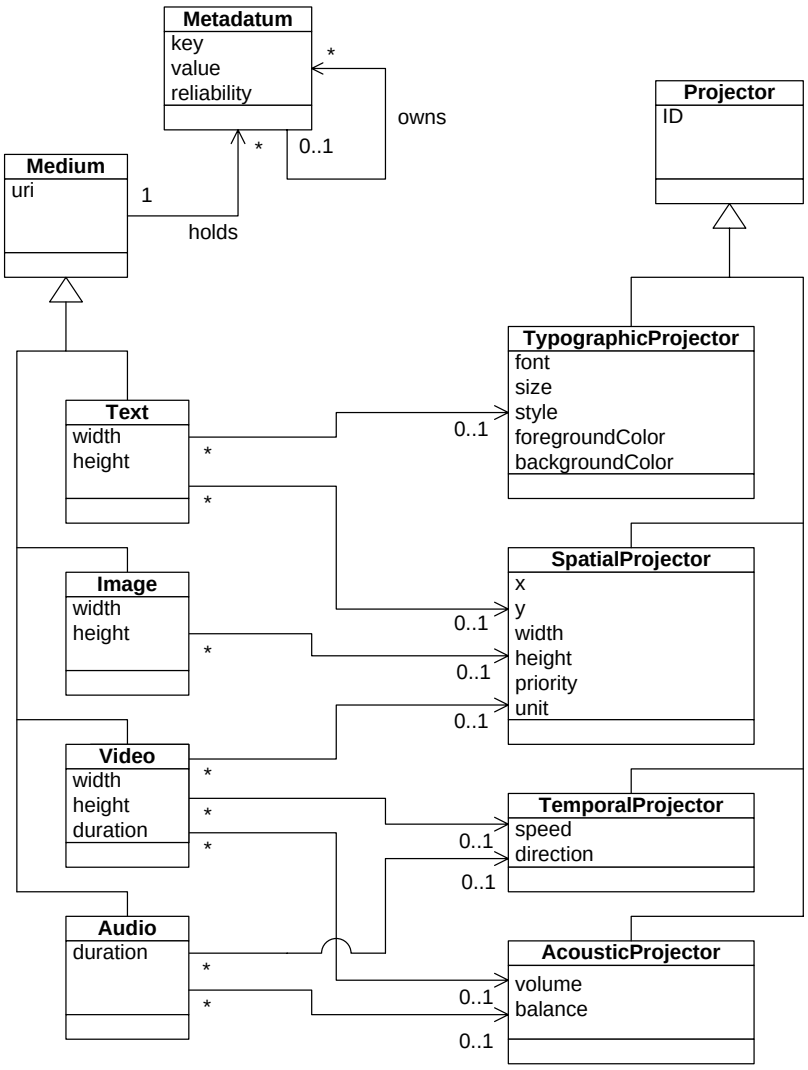


Figure B.2: The media elements and projectors of the internal multimedia content representation model

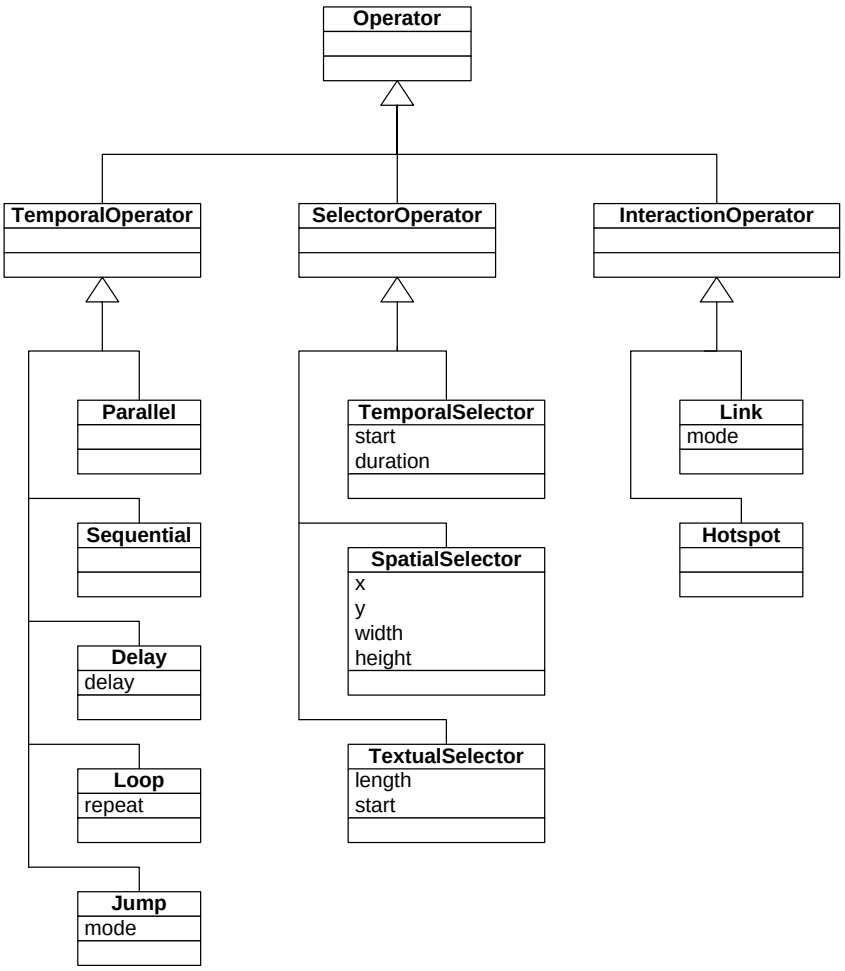


Figure B.3: The hierarchy of the basic composition operators

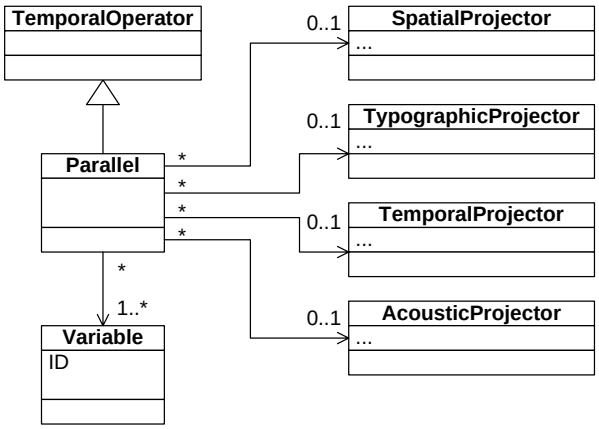


Figure B.4: The basic temporal operator Parallel

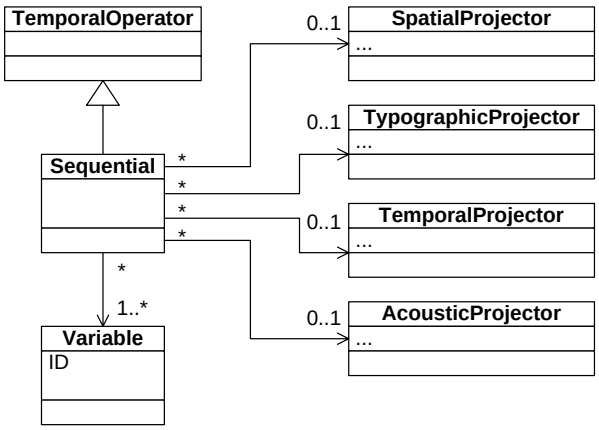


Figure B.5: The basic temporal operator Sequential

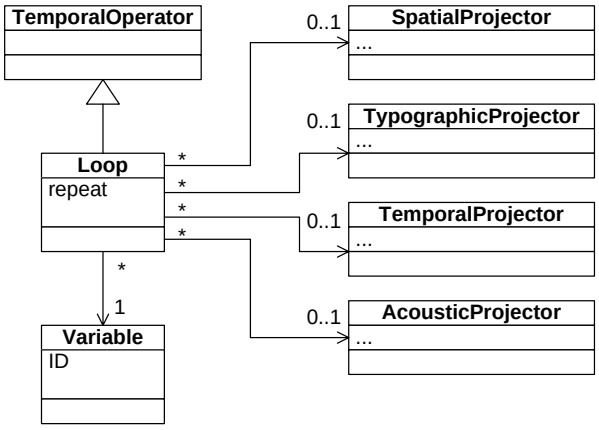


Figure B.6: The basic temporal operator Loop

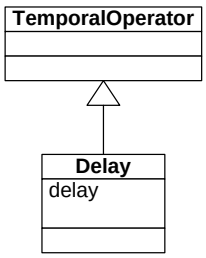


Figure B.7: The basic temporal operator Delay

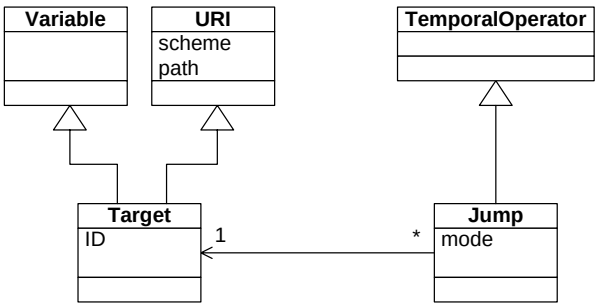


Figure B.8: The basic temporal operator Jump

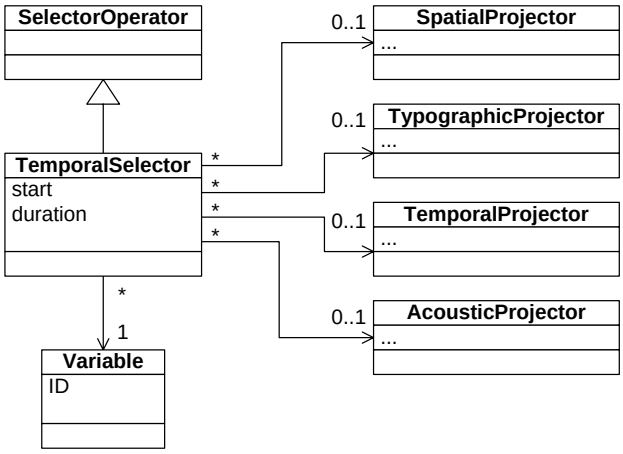


Figure B.9: The basic selector operator TemporalSelector

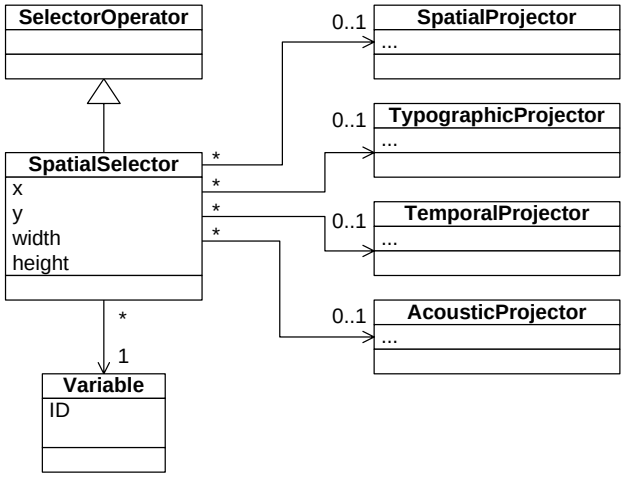


Figure B.10: The basic selector operator SpatialSelector

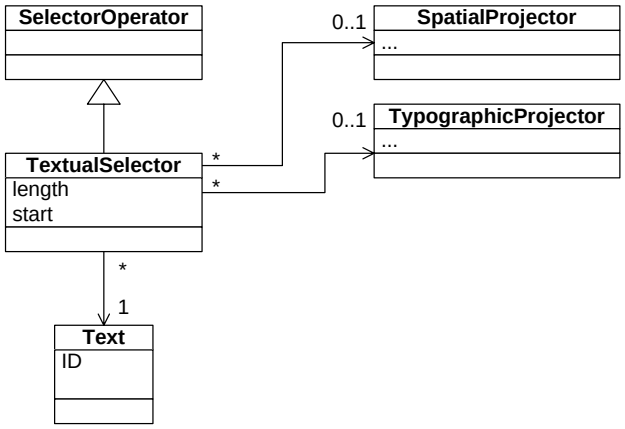


Figure B.11: The basic selector operator TextualSelector

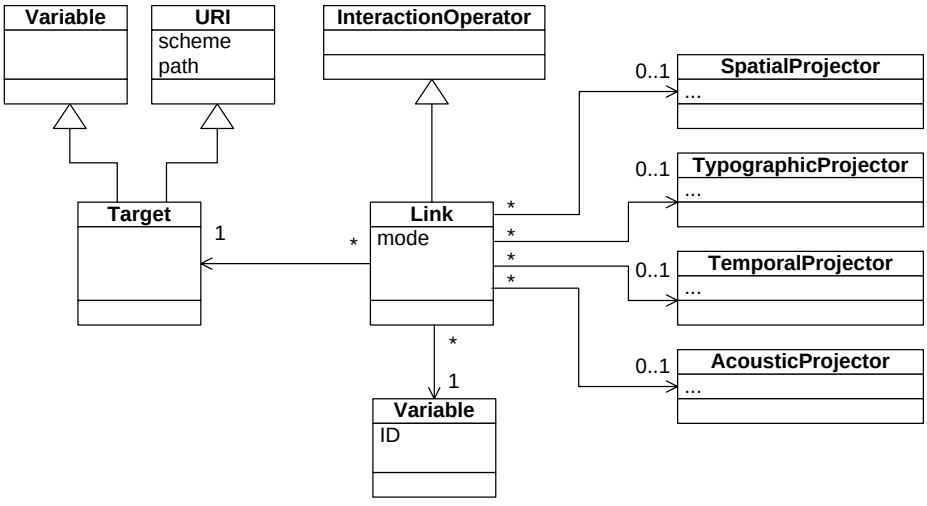


Figure B.12: The basic interaction operator Link

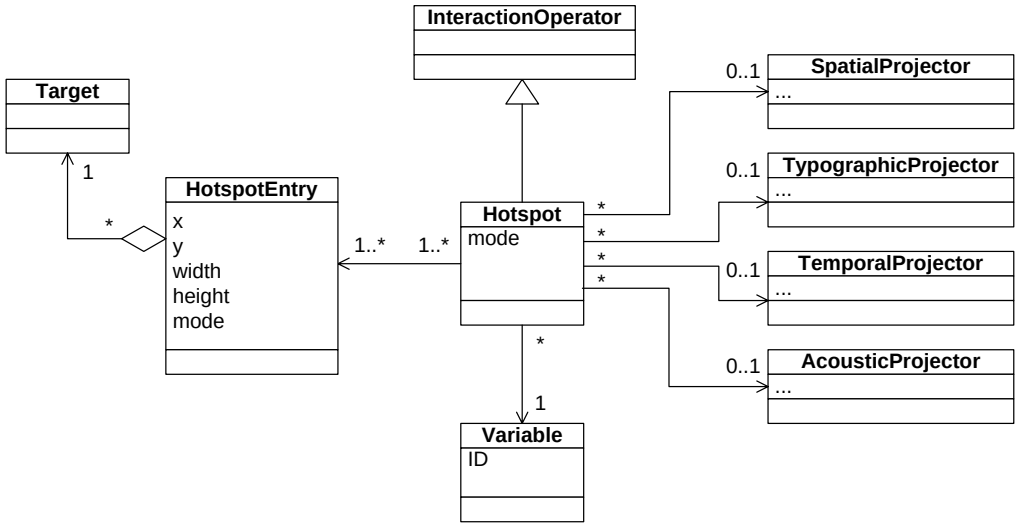


Figure B.13: The basic interaction operator Hotspot

C DTD of the Internal Multimedia Content Representation Model

The following Listing C.1 shows the XML DTD for representing multimedia content provided as input for the Transformation4U application presented in Section 10.4. It comprises a XML DTD specification of all basic composition operators, media elements, and projectors as defined for the internal multimedia content representation model in Section 6.3.3.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT MM4U (Header?, Body)>
<!ATTLIST MM4U version CDATA #REQUIRED>

<!-- Header -->
<!ELEMENT Header (Layout?)>
<!ATTLIST Header title CDATA #IMPLIED>

<!ELEMENT Layout (SpatialProjector|AcousticProjector|TypographicProjector|
    TemporalProjector)*>
<!ATTLIST Layout
    width CDATA #IMPLIED
    height CDATA #IMPLIED
>

<!ELEMENT SpatialProjector (SpatialProjector|AcousticProjector|
    TypographicProjector|TemporalProjector)*>
<!ATTLIST SpatialProjector
    id ID #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    priority CDATA #REQUIRED
    unit CDATA #IMPLIED
>

<!ELEMENT AcousticProjector (SpatialProjector|AcousticProjector|
    TypographicProjector|TemporalProjector)*>
<!ATTLIST AcousticProjector
    id ID #REQUIRED
    volume CDATA #IMPLIED
    balance CDATA #IMPLIED
>

<!ELEMENT TypographicProjector (SpatialProjector|AcousticProjector|
    TypographicProjector|TemporalProjector)*>
<!ATTLIST TypographicProjector
    id ID #REQUIRED
```

```
    font CDATA #IMPLIED
    size CDATA #IMPLIED
    style CDATA #IMPLIED
    foregroundColor CDATA #IMPLIED
    backgroundColor CDATA #IMPLIED
>

<!ELEMENT TemporalProjector (SpatialProjector|AcousticProjector|
    TypographicProjector|TemporalProjector)*>
<!ATTLIST TemporalProjector
    id ID #REQUIRED
    speed CDATA #IMPLIED
    direction CDATA #IMPLIED
>

<!— Body —>
<!ELEMENT Body (Image|Text|Audio|Video|Sequential|Parallel|Link|Jump|Loop|
    Hotspot|TemporalSelector|SpatialSelector|TextualSelector)>

<!ELEMENT Image (ProjectorList?)>
<!ATTLIST Image
    id ID #REQUIRED
    uri CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    keywords CDATA #IMPLIED
    description CDATA #IMPLIED
>

<!ELEMENT Audio (ProjectorList?)>
<!ATTLIST Audio
    id ID #REQUIRED
    uri CDATA #REQUIRED
    duration CDATA #IMPLIED
    keywords CDATA #IMPLIED
    description CDATA #IMPLIED
>

<!ELEMENT Video (ProjectorList?)>
<!ATTLIST Video
    id ID #REQUIRED
    uri CDATA #REQUIRED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    duration CDATA #IMPLIED
    keywords CDATA #IMPLIED
    description CDATA #IMPLIED
>

<!ELEMENT Text (ProjectorList?)>
<!ATTLIST Text
  id ID #REQUIRED
  uri CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA #REQUIRED
  content CDATA #IMPLIED
  keywords CDATA #IMPLIED
  description CDATA #IMPLIED
>
```

```
<!— Projector and list of projectors —>
<!ELEMENT Projector EMPTY>
<!ATTLIST Projector refid IDREF #REQUIRED>

<!ELEMENT ProjectorList (Projector)*>
<!ATTLIST ProjectorList id CDATA #IMPLIED>

<!— Basic composition operators —>
<!ELEMENT Sequential
    ( (Sequential|Parallel|Link|Jump|Loop|Hotspot|TemporalSelector|
       SpatialSelector|TextualSelector|Image|Text|Audio|Video)*,
       ProjectorList?)>
<!ATTLIST Sequential id ID #REQUIRED>

<!ELEMENT Parallel
    ( (Sequential|Parallel|Link|Jump|Loop|Hotspot|TemporalSelector|
       SpatialSelector|TextualSelector|Image|Text|Audio|Video)*,
       ProjectorList?)>
<!ATTLIST Parallel id ID #REQUIRED>

<!ELEMENT Link
    ( (Sequential|Parallel|Loop|Hotspot|TemporalSelector|SpatialSelector|
       TextualSelector|Image|Text|Audio|Video), ProjectorList?)>
<!ATTLIST Link
    id ID #REQUIRED
    target CDATA #REQUIRED
    mode (stop|spawn) #IMPLIED
>

<!ELEMENT Jump EMPTY>
<!ATTLIST Jump
    id ID #REQUIRED
    target CDATA #REQUIRED
    mode (stop|spawn) #IMPLIED
>

<!ELEMENT Loop
    ( (Sequential|Parallel|Link|Loop|Hotspot|TemporalSelector|SpatialSelector|
       TextualSelector|Image|Text|Audio|Video), ProjectorList?)>
<!ATTLIST Loop
    id ID #REQUIRED
    repeat CDATA #REQUIRED
>

<!ELEMENT TemporalSelector
    ( (Sequential|Parallel|Link|Loop|Hotspot|TemporalSelector|SpatialSelector|
       TextualSelector|Image|Text|Audio|Video), ProjectorList?)>
<!ATTLIST TemporalSelector
    id ID #REQUIRED
    start CDATA #REQUIRED
    duration CDATA #REQUIRED
>

<!ELEMENT SpatialSelector
    ( (Sequential|Parallel|Link|Loop|Hotspot|TemporalSelector|SpatialSelector|
       TextualSelector|Image|Text|Audio|Video), ProjectorList?)>
<!ATTLIST SpatialSelector
    id ID #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
```

```
160     width CDATA #REQUIRED
        height CDATA #REQUIRED
    >

    <!ELEMENT TextualSelector (Text, ProjectorList?)>
165 <!ATTLIST TextualSelector
        id ID #REQUIRED
        start CDATA #REQUIRED
        length CDATA #REQUIRED
    >
170
    <!ELEMENT Hotspot
         ( (TemporalSelector|SpatialSelector|TextualSelector|Parallel|Sequential|
           Loop|Hotspot|Image|Text|Audio|Video), HotspotEntry*,
           ProjectorList?)>
175 <!ATTLIST Hotspot
        id ID #REQUIRED
    >

    <!ELEMENT HotspotEntry EMPTY>
180 <!ATTLIST HotspotEntry
        x CDATA #REQUIRED
        y CDATA #REQUIRED
        width CDATA #REQUIRED
        height CDATA #REQUIRED
185     target CDATA #REQUIRED
        mode (stop | spawn) #IMPLIED
    >
```

Listing C.1: XML DTD of the internal multimedia content representation model

D DTD of the Flash Markup Language

The following Listing D.1 shows the XML DTD of the Flash Markup Language (FML). For a detailed description of the FML we refer the reader to Section 6.4.7.2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT FlashMovie (Header, Define, Body)>

<!-- Header -->
<!ELEMENT Header (FrameSize, BackgroundColor)>
<!ATTLIST Header
    flashVersion CDATA #REQUIRED
    frameRate CDATA #REQUIRED
>

<!ELEMENT FrameSize EMPTY>
<!ATTLIST FrameSize
    width CDATA #REQUIRED
    height CDATA #REQUIRED
>

<!ELEMENT BackgroundColor EMPTY>
<!ATTLIST BackgroundColor
    red CDATA #REQUIRED
    green CDATA #REQUIRED
    blue CDATA #REQUIRED
>

<!ELEMENT Define (DefineSound|DefineImage|DefineText|DefineVideo|DefineLoop)*>

<!ELEMENT DefineImage EMPTY>
<!ATTLIST DefineImage
    characterID CDATA #REQUIRED
    imageID CDATA #REQUIRED
    src CDATA #REQUIRED
    width CDATA #REQUIRED
    height CDATA #REQUIRED
    rotate CDATA #IMPLIED
>

<!ELEMENT DefineText ANY>
<!ATTLIST DefineText
    characterID CDATA #REQUIRED
    width CDATA #REQUIRED
    height CDATA #REQUIRED
    rotate CDATA #IMPLIED
>

<!ELEMENT DefineSound EMPTY>
<!ATTLIST DefineSound
```

```
    characterID CDATA #REQUIRED
    src CDATA #REQUIRED
    clipBegin CDATA #IMPLIED
>

<!ELEMENT DefineVideo EMPTY>
<!ATTLIST DefineVideo
    characterID CDATA #REQUIRED
    src CDATA #REQUIRED
    width CDATA #REQUIRED
    height CDATA #REQUIRED
    rotate CDATA #IMPLIED
    clipBegin CDATA #IMPLIED
>

<!ELEMENT DefineLoop EMPTY>
<!ATTLIST DefineLoop
    loopName CDATA #REQUIRED
>

<!— Body —>
<!ELEMENT Body (ShowFrame)*>

<!ELEMENT ShowFrame (PlaceObject|RemoveObject|StartSound|StopSound|JumpTo|
    LoopTo)*>

<!ATTLIST ShowFrame
    name CDATA #IMPLIED
    duration CDATA #IMPLIED
>

<!ELEMENT PlaceObject (Hotspot)*>
<!ATTLIST PlaceObject
    characterIDRef CDATA #REQUIRED
    depth CDATA #REQUIRED
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    target CDATA #IMPLIED
    mode CDATA #IMPLIED
>

<!ELEMENT Hotspot EMPTY>
<!ATTLIST Hotspot
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    width CDATA #REQUIRED
    height CDATA #REQUIRED
    target CDATA #REQUIRED
    mode (stop | spawn) #IMPLIED
>

<!ELEMENT RemoveObject EMPTY>
<!ATTLIST RemoveObject
    depth CDATA #REQUIRED
>

<!ELEMENT StartSound EMPTY>
<!ATTLIST StartSound
    characterIDRef CDATA #REQUIRED
    volume CDATA #REQUIRED
```

```
    balance CDATA #REQUIRED
>

<!ELEMENT StopSound EMPTY>
<!ATTLIST StopSound
    characterIDRef CDATA #REQUIRED
>

<!ELEMENT JumpTo EMPTY>
<!ATTLIST JumpTo
    target CDATA #REQUIRED
    mode CDATA #IMPLIED
>

<!ELEMENT LoopTo EMPTY>
<!ATTLIST LoopTo
    target CDATA #REQUIRED
    repeat CDATA #REQUIRED
>

<!— Define basic XHTML text formatting elements —>
<!ELEMENT br EMPTY>

<!ELEMENT b ANY>
<!ELEMENT u ANY>
<!ELEMENT i ANY>

<!ELEMENT font ANY>
<!ATTLIST font
    size  CDATA #IMPLIED
    color CDATA #IMPLIED
    face  CDATA #IMPLIED
>
```

Listing D.1: XML DTD of the Flash Markup Language

Glossary

C

Component See ↑software component.

Context See ↑user model.

Contract A contract of a ↑software component guaranties in case of calling a service "as defined per contract" that the execution of a specific action is conducted "as defined in the contract". A component contract includes a specification of all functional as well as non-functional characteristics of the services it offers. What services are offered by a component is specified in the provided interface. Thus, the provided interface determines the postconditions of a component. The services and components needed by a component to fulfill its own functionality is determined in the required interface. It defines the preconditions of a component. Concerning the component contract, this means provided that the preconditions of a component are fulfilled by offering the right execution environment, the component offers its services as described in the postconditions. As the preconditions of a component might not be fully satisfied by a specific execution environment and the postconditions of a component are not necessary to their full extend by all components using it, also subsets of the preconditions and postconditions can be modeled in the component's contract. Such a component contract that can be adapted to different specific execution environments of a component is called parameterized contract.

Customization Customization is an activity that is conducted and under direct control by the user. This means that a customized application is actively adapted by the users to their individual needs and requirements. However, the customized application does not provide to adapt itself to the users. In contrast to ↑personalization, where the provided information, services, and products are adapted by the personalized application itself according to the user profile information.

F

Framework See ↑software framework.

Frozen-spot Part of a ↑software framework that cannot be modified by the concrete application.

G

Group-hot-spot A group-hot-spot is a set of ↑hot-spots that are logically coherent. This means, that the set of hot-spots describes flexibility requirements for a common problem in the framework's domain and thus the hot-spots have a high affinity in regard of solving that particular problem.

H

Hot-spot Part of a ↑software framework where adaptation towards the requirements of a concrete application is possible.

I

Interface See ↑provided interface and ↑required interface of a ↑software component.

M

Medium A medium is determined by its ↑medium type. It can be either continuous or discrete.

Medium Element A medium element is an instance of a specific ↑medium type. Consequently, a medium element can be either continuous or discrete.

Medium Format A medium format is a defined data structure for a specific ↑medium element. This data structure encodes the medium element.

Medium Type A medium type determines the type of a ↑medium element. There exist medium types for continuous media elements such as audio, video, and animation. Medium types for discrete media elements are image and text.

Meta Data Meta data are considered as additional data about data. They are describing information about data in order to provide for finding and using the data more efficiently.

Multimedia Multimedia is characterized by the computer-controlled, integrated creation, manipulation (i. e., interaction of the user with the media), presentation, storage, and communication of independent information. This independent (multimedia) information is encoded at least through one continuous and one discrete ↑medium element.

Multimedia Content Authoring Multimedia authoring is the process in which ↑multimedia presentations are created.

Multimedia Composition See ↑multimedia content.

Multimedia Content Multimedia content is seen as the result of a composition of different ↑media elements such as images, text, audio, and video into an interactive continuous ↑multimedia presentation, comprising at least one discrete and one continuous medium element. Central features of such a multimedia composition are the temporal arrangement of the media elements in the course of the presentation, the layout of the presentation, and the definition of the presentation's interaction possibilities with the users.

Multimedia Content Representation See ↑multimedia document.

Multimedia Document Representation of ↑multimedia content. It is an instantiation of a ↑multimedia document model. A multimedia document that is composed in advance to its rendering is called pre-orchestrated in contrast to compositions that take place just before rendering that are called live or on-the-fly.

Multimedia Document Representation See ↑multimedia presentation format.

Multimedia Document Format See ↑multimedia presentation format.

Multimedia Document Model A multimedia document model provides the primitives to capture all aspects of a ↑multimedia document. The power of the multimedia document model determines the degree of the multimedia functionality that multimedia documents following the model can provide.

Multimedia Player A multimedia player is a multimedia software that is aimed at the rendering and playback of ↑multimedia presentations.

Multimedia Presentation Format A multimedia presentation format determines the representation of a ↑multimedia document for the document's rendering. Since every multimedia presentation format implicitly or explicitly follows a ↑multimedia document model, it can also be seen as a proper mean to "serialize" the multimedia document's representation for the purpose of exchange.

Multimedia Presentation A multimedia presentation is the rendering of a ↑multimedia document in a concrete ↑multimedia presentation format by an appropriate ↑multimedia player. It comprises the continuous rendering of the document in the target environment, the (pre)loading of media data, realizing the temporal course, the temporal synchronization between continuous media streams, the adaptation to different or changing presentation conditions, and the interaction with the user.

P

Personalization Personalization is defined as the offer and opportunity for a special treatment in form of information, services, and products, provided by an application according to the interests, background, role, facts, requirements, needs, and any other information and assumptions about an individual. The personalization of these information, services, and products is conducted pro-actively by the personalized application (in contrast to ↑customization) and is typically carried out to the user in an iterative process.

Personalized Multimedia Content ↑Multimedia content that is targeted at a specific user or user group. It reflects the individual user's or user group's needs, background, interest, and knowledge, as well as the heterogeneous infrastructure of the (mobile) end device to which the personalized multimedia content is delivered and on which it is presented (see also ↑personalization).

Personalized Multimedia Content Authoring Process of dynamically selecting and composing media elements into a coherent, continuous multimedia presentation (see ↑multimedia content authoring) that best reflects the needs, requirements, and system environment of an individual user or user group, i. e., into a ↑personalized multimedia presentation.

Provided Interface The provided interface specifies the services offered by a ↑software component.

R

Realization The realization determines the inside of a ↑software component, i. e., the component's internal, private design and implementation.

Required Interface The required interface determines the services and components needed by a ↑software component to fulfill its functionality, i. e., its ↑provided interface.

S

Software Component Software components are modular units of a software system that encapsulate their content and thus their internal (complex) behavior (see ↑realization). They appear to the environment as exchangeable units and interact with the environment via well-defined (contractually specified) interfaces as defined in the component's ↑contract.

Software Framework A software framework typically constitutes a semi-finished software architecture for a complex application domain that can be adapted to the needs and requirements of a concrete application in the domain by its ↑variation points. We distinguish between object-oriented frameworks and component frameworks. These kinds of frameworks differ in the technology used for adapting and specializing the frameworks to the needs and requirements of the concrete applications. In case of object-oriented frameworks, the adaptation and specialization for a concrete application is conducted by overriding abstract classes or by different implementations of predefined interfaces. With component frameworks, specialization takes place by composing different ↑software components that implement the component framework's interfaces.

U

User Model A user model describes an abstract data structure. This data structure defines what kind of information and assumptions about a user and his or her environment and situation is modeled and processed by a personalized software system (see ↑personalization). The user model's data structure includes the aspects of both classical user modeling and context modeling, i. e., classical user profile information and context information.

User Profile In contrast to the ↑user model, which defines an abstract data structure, a user profile contains the actual information about an individual user of a concrete personalized application (see ↑personalization). Hence, user profiles can be seen as "instances" of a user model, filled in with the concrete values about an individual user.

V

Variation Point Variation points are those parts in the ↑software framework's architecture, where the decision for a specific functionality is already defined. However, the actual implementation of this functionality is left open and will be determined by the concrete applications employing the framework.

List of Figures

List of Tables

List of Listings

List of Abbreviations

3GPP	3rd Generation Partnership Project
AHM	Amsterdam Hypermedia Model
CC/PP	Composite Capability/Preference Profiles
CSS	Cascading Style Sheets
DTD	Document Type Definition
FML	Flash Markup Language
FSA	Finite State Automaton
HFO	Hypermedia Formatting Objects
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IMMPS	Intelligent Multimedia Presentation System
ISO	International Organization for Standardization
LASeR	Lightweight Applications Scene Representation
MM4U	MultiMedia For You
PDA	Personal Digital Assistant
POI	Point of Interest
ProMoCF	Process Model for Component Frameworks
SMIL	Synchronized Multimedia Integration Language
SMIL BLP	SMIL Basic Language Profile
SRM for IMMPS	Standard Reference Model for IMMPSs
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XMT	Extensible MPEG-4 Textual
XSL	Extensible Stylesheet Language
XSL-FO	XSL Formatting Objects
XSLT	XSL Transformations
W3C	World Wide Web Consortium

Bibliography

[3rd03a] 3RD GENERATION PARTNERSHIP PROJECT: TS 26.234; Transparent end-to-end Packet-switched Streaming Service; Protocols and codecs (Release 5). 2003. Retrieved March 6, 2006, from http://www.3gpp.org/ftp/Specs/html-info/26234.htm

[3rd03b] 3RD GENERATION PARTNERSHIP PROJECT: TS 26.246; Transparent end-to-end Packet-switched Streaming Service; 3GPP SMIL language profile (Release 6). 2003. Retrieved March 6, 2006, from http://www.3gpp.org/ftp/Specs/html-info/26246.htm

[AAA+04a] ANDERSSON, O.; AXELSSON, H.; ARMSTRONG, P.; BALCISOY, S.; ET AL.: Mobile SVG Profiles: SVG Tiny and SVG Basic - W3C Recommendation. Mar. 2004. Retrieved March 6, 2006, from http://www.w3.org/TR/SVGMobile12/

[AAA+04b] ANDERSSON, O.; AXELSSON, H.; ARMSTRONG, P.; BALCISOY, S.; ET AL.: Scalable Vector Graphics (SVG) 1.2 Specification: W3C Working Draft. Oct. 2004. Retrieved March 6, 2006, from http://www.w3.org/TR/2004/WD-SVG12-20040510/

[ABB+02] ATKINSON, C.; BAYER, J.; BUNSE, C.; KAMSTIES, E.; LAITENBERGER, O.; LAQUA, R.; MUTHIG, D.; PAECH, B.; WÜST, J.; ZETTEL, J.: *Component-based product line engineering with UML*. Addison-Wesley, 2002

[ABC+01a] ADLER, S.; BERGLUND, A.; CARUSO, J.; DEACH, S.; GRAHAM, T.; GROSSO, P.; GUTENTAG, E.; MILOWSKI, A.; PARNELL, S.; RICHMAN, J.; ZILLES, S.: Extensible Stylesheet Language (XSL) Version 1.0: W3C Recommendation. Oct. 2001. Retrieved March 22, 2006, from http://www.w3.org/TR/2001/REC-xsl-20011015/xslspecRX.pdf

[ABC+01b] AYARS, J.; BULTERMAN, D.; COHEN, A.; DAY, K.; HODGE, E.; HOSCHKA, P.; ET AL.: Synchronized Multimedia Integration Language (SMIL 2.0) Specification - W3C Recommendation. Aug. 2001. Retrieved March 6, 2006, from http://www.w3.org/TR/2001/REC-smil20-20010807/

[ABDG02] AÏMEUR, E.; BRASSARD, G.; DUFORT, H.; GAMBS, S.: CLARISSE:A Machine Learning Tool to Initialize Student Models. In: CERRI, S. A.; GOUARDÈRES, G.; PARAGUAÇU, F. (ed.), *Proc. of the 6th Int. Conf. on Intelligent Tutoring Systems; Biarritz, France and San Sebastian, Spain*, Springer-Verlag, Jun. 2002, pp. 718–728

[ABH97] APERS, P. M. G.; BLANKEN, H. M.; HOUTSMA, M. A.: *Multimedia Databases in Perspective*. Springer-Verlag, 1997

[Ack96] ACKERMANN, P.: *Developing Object-Oriented Multimedia Software: Based on MET++ Application Framework*. dpunkt.verlag, 1996

[Ado05a] ADOBE SYSTEMS, INC., USA: Macromedia - Director MX 2004. 2005. Retrieved March 6, 2006, from http://www.macromedia.com/software/director/

[Ado05b] ADOBE SYSTEMS, INC., USA: Using Authorware 7 [Computer manual]. Jan. 2005. Available from http://www.macromedia.com/software/authorware/

[Ado06a] ADOBE SYSTEMS, INC., USA: Adobe SVG Viewer. 2006. Retrieved March 6, 2006, from http://www.adobe.com/svg/

[Ado06b] ADOBE SYSTEMS, INC., USA: Flash Professional 8. 2006. Retrieved March 20, 2006 from http://www.macromedia.com/software/flash/flashpro/

[Ado06c] ADOBE SYSTEMS, INC., USA: Macromedia Flash Player. 2006. Retrieved March 6, 2006, from http://www.macromedia.com/go/getflashplayer

[Ado06d] ADOBE SYSTEMS, INC., USA: Macromedia Flash Player Penetration. Sept. 2006. Retrieved April 14, 2006, from http://www.macromedia.com/software/player_census/flashplayer/penetration.html

[Ado06e] ADOBE SYSTEMS, INC., USA: Macromedia Flex. 2006. Retrieved February 8, 2006, from http://www.macromedia.com/software/flex/

[AFG+93] ANDRÉ, E.; FINKLER, W.; GRAF, W.; RIST, T.; SHAUDER, A.; WAHLSTER, W.: WIP: the automatic synthesis of multimodal presentations. In: MAYBURY [May93], ch. 3, pp. 75–93

[AGPS02] ARDISSONO, L.; GOY, A.; PETRONE, G.; SEGNAN, M.: Personalization in business-to-customer interaction. In: *Communications of the ACM* 45 (2002), no. 5, pp. 52–53

[All83] ALLEN, J. F.: Maintaining knowledge about temporal intervals. In: *Communications of the ACM* 26 (1983), no. 11, pp. 832–843

[All90] ALLEN, R. B.: User models: theory, method, and practice. In: *Int. Journal of Man-Machine Studies* 32 (1990), no. 5, pp. 511–543

[All99] ALLEN, C.: Personalization vs. Customization. Jul. 1999. Retrieved February 28, 2006, from http://www.clickz.com/experts/archives/mkt/precis_mkt/article.php/814811

[ALRL04] AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B.; LANDWEHR, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. In: *IEEE Trans. Dependable Secur. Comput.* 1 (2004), no. 1, pp. 11–33

[Ama06] AMAZON, INC, USA: amazon.com. 2006. Retrieved March 6, 2006, from http://www.amazon.com/

[AMR96a] ANDRÉ, E.; MÜLLER, J.; RIST, T.: WIP/PPP: automatic generation of personalized multimedia presentations. In: *Proc. of the 4th ACM Int. Conf. on Multimedia; Boston, MA, USA*, ACM Press, 1996, pp. 407–408

[AMR96b] ANDRÉ, E.; MÜLLER, J.; RIST, T.: WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring. In: *Proc. of EUROMEDIA; London, UK*, 1996, pp. 95–102

[Ams02a] AMSTERDAM SCIENCE & TECHNOLOGY CENTRE, THE NETHERLANDS: Multimedia Information Analysis: The value of networked vision. 2002

[Ams02b] AMSTERDAM SCIENCE & TECHNOLOGY CENTRE, THE NETHERLANDS: Virtual Laboratory: A toolkit for sharing resources. 2002

[Ang03] ANGELIDES, M. C.: Guest Editor's Introduction: Multimedia Content Modeling and Personalization. In: *IEEE Multimedia* 10 (2003), no. 4, pp. 12–15

[App05] APPLE, USA: QuickTime. 2005. Retrieved March 24, 2006, from http://www.apple.com/quicktime/

[App06] APPLE, USA: QuickTime Documentation. 2006. Retrieved February 7, 2006, from http://developer.apple.com/documentation/QuickTime/

[AR95] ANDRÉ, E.; RIST, T.: Generating coherent presentations employing textual and visual material. In: *Artif. Intell. Rev.* 9 (1995), no. 2-3, pp. 147–165

[AR96] ANDRÉ, E.; RIST, T.: Coping with Temporal Constraints in Multimedia Presentation Planning. In: *Proc. of the 13th National Conf. on Artificial Intelligence; Portland, OR, USA*, Aug. 1996, pp. 142–147

[ASS99] ADALI, S.; SAPINO, M. L.; SUBRAHMANIAN, V. S.: A multimedia presentation algebra. In: *Proc. of the 1999 ACM SIGMOD Int. Conf. on Management of data; Philadelphia, PA, USA*, ACM Press, 1999, pp. 121–132

[ASS00] ADALI, S.; SAPINO, M. L.; SUBRAHMANIAN, V. S.: An algebra for creating and querying multimedia presentations. In: *Multimedia Syst.* 8 (2000), no. 3, pp. 212–230

[BB99] BENTSSON, P.-O.; BOSCH, J.: Haemo Dialysis Software Architecture Design Experiences. In: *Proc. of the 21st Int. Conf. on Software Engineering*, May 1999, pp. 516–526

[BBB+98] BUGAJ, S.; BULTERMAN, D.; BUTTERFIELD, B.; CHANG, W.; FOUQUET, G.; GRAN, C.; ET AL.: Synchronized Multimedia Integration Language (SMIL 1.0) Specification: W3C Recommandation. Jun. 1998. Retrieved March 6, 2006, from http://www.w3.org/TR/REC-smil/

[BBK+04] BALDZER, J.; BOLL, S.; KLANTE, P.; KRÖSCHE, J.; MEYER, J.; RUMP, N.; SCHERP, A.: Location-Aware Mobile Multimedia Applications on the Niccimon Platform. In: *Informationssysteme für mobile Anwendungen; Brunswick, Germany*, Oct. 2004, pp. 318–334

[BC89] BECK, K.; CUNNINGHAM, W.: A laboratory for teaching object oriented thinking. In: *ACM SIGPLAN Notices* 24 (1989), no. 10, pp. 1–6. Conference on Object-Oriented Programming: Systems, Languages and Applications; New Orleans, LA, USA

[BC01] BYUN, H. E.; CHEVERST, K.: Exploiting User Models and Context-Awareness to Support Personal Daily Activities. In: *Workshop in UM2001 on User Modelling for Context-Aware Applications; Sonthofen, Germany*, 2001

[BC02] BENTAL, D.; CAWSEY, A.: Personalized and adaptive systems for medical consumer applications. In: *Communications of the ACM* 45 (2002), no. 5, pp. 62–63

[BD00] BRUEGGE, B.; DUTOIT, A. H.: *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. Prentice Hall, 2000

[BDLN04] BADE, K.; DE LUCA, E. W.; NÜRNBERGER, A.: Multimedia Retrieval: Fundamental Techniques and Principles of Adaptivity. In: *KI* 18 (2004), no. 4, pp. 5–10

[BE98] BOYLE, C.; ENCARNACION, A. O.: Metadoc: An Adaptive Hypertext Reading System. In: BRUSILOVSKY, P.; KOBSA, A.; VASSILEVA, J. (ed.), *Adaptive Hypertext and Hypermedia*, Kluwer Academic Publishers, 1998, pp. 71–89

[Bec00] BECK, K.: *Extreme programming explained: embrace design*. Addison-Wesley, Dec. 2000

[BFF+97] BORDEGONI, M.; FACONTI, G.; FEINER, S.; MAYBURY, M. T.; RIST, T.; RUGGIERI, S.; TRAHANIAS, P.; WILSON, M.: A standard reference model for intelligent multimedia presentation systems. In: *Computer Standards & Interfaces* 18 (1997), no. 6-7, pp. 477–496

[BFR+96] BORDEGONI, M.; FACONTI, G.; RIST, T.; RUGGIERI, S.; TRAHANIAS, P.; WILSON, M.: Intelligent Multimedia Presentation Systems: A Proposal for a Reference Model. In: *The Int. Conf. on Multi-Media Modeling; Toulouse, France*, Nov. 1996, pp. 3–20

[BH00] BOHRER, K.; HOLLAND, B.: Customer Profile Exchange (CPExchange) Specification - Version 1.0, 20 October 2000. 2000. Retrieved March 6, 2006, from http://www.cpexchange.org/specification/cpexchangev1_0F.pdf

[BH05] BULTERMAN, D. C. A.; HARDMAN, L.: Structured multimedia authoring. In: *ACM Trans. on Multimedia Computing, Communications, and Applications* 1 (2005), no. 1, pp. 89–109

[BHJ+98] BULTERMAN, D. C. A.; HARDMAN, L.; JANSEN, J.; MULLENDER, K. S.; RUTLEDGE, L.: GRiNS: a graphical interface for creating and playing SMIL documents. In: *Proc. of the 7th Int. Conf. on World Wide Web; Brisbane, Australia*, Elsevier, 1998, pp. 519–529

[BIM+00] BAKER, M.; ISHIKAWA, M.; MATSUI, S.; STARK, P.; WUGOFSKI, T.; YAMAKAMI, T.: XHTML Basic: W3C Recommendation. Dec. 2000. Retrieved February 8, 2006, from http://www.w3.org/TR/xhtml-basic/

[BJK01] BES, F.; JOURDAN, M.; KHANTACHE, F.: A Generic Architecture for Automated Construction of Multimedia Presentations. In: *Int. Conf. on Multimedia Modeling; Amsterdam, The Netherlands*, Nov. 2001

[BJK+04] BULTERMAN, D. C. A.; JANSEN, J.; KLEANTHOUS, K.; BLOM, K.; BENDEN, D.: Ambulant: a fast, multi-platform open source SMIL player. In: *Proc. of the 12th annual ACM Int. Conf. on Multimedia; New York, NY, USA*, ACM Press, 2004, pp. 492–495

[BJSF94] BEARNE, M.; JONES, S.; SAPSFORD-FRANCIS, J.: Towards usability guidelines for multimedia systems. In: *Proc. of the 2nd ACM Int. Conf. on Multimedia; San Francisco, CA, USA*, ACM Press, 1994, pp. 105–110

[BK97] BAILEY, B.; KONSTAN, J. A.: Nsync: A Constraint Based Toolkit for Multimedia. In: *Proc. of the 5th Annual Tcl/Tk Workshop; Boston, MA, USA*, Usenix Association, Berkeley, Jul. 1997, pp. 107–114

[BK01] BOLL, S.; KLAS, W.: ZYX - A Multimedia Document Model for Reuse and Adaptation. In: *IEEE Trans. on Knowledge and Data Engineering* 13 (2001), no. 3, pp. 361–382

[BK03] BLOCK, M.; KONRAD, S.: *Personalized and multimedia Webservice: Sightseeing4U*. Individual projects, Carl von Ossietzky University Oldenburg, Department of Computing Science, Oldenburg, Germany, Sept. 2003

[BKCD98] BAILEY, B.; KONSTAN, J. A.; COOLEY, R.; DEJONG, M.: Nsync - a toolkit for building interactive multimedia presentations. In: *Proc. of the 6th ACM Int. Conf. on Multimedia; Bristol, UK*, ACM Press, 1998, pp. 257–266

[BKHW01] BOLL, S.; KLAS, W.; HEINLEIN, C.; WESTERMANN, U.: *Cardio-OP: Anatomy of a Multimedia Repository for Cardiac Surgery*. tech. rep. TR-2001301, University of Vienna, Austria, Aug. 2001

[BKL96] BOLL, S.; KLAS, W.; LÖHR, M.: Integrated Database Services for Multimedia Presentations. In: CHUNG, S. M. (ed.), *Multimedia Information Storage and Management*, Kluwer Academic Publishers, ch. 16, Aug. 1996

[BKS04] BOLL, S.; KRÖSCHE, J.; SCHERP, A.: Personalized Multimedia meets Location-Based Services. In: *Workshop Multimedia-Informationssysteme, Informatik 2004; Ulm, Germany*, Gesellschaft für Informatik, Germany, Sept. 2004

[BKW99a] BOLL, S.; KLAS, W.; WANDEL, J.: A Cross-Media Adaptation Strategy for Multimedia Presentations. In: *Proc. of the ACM Multimedia Conf., Part 1; Orlando, FL, USA*, Nov. 1999, pp. 37–46

[BKW99b] BOLL, S.; KLAS, W.; WESTERMANN, U.: Multimedia Document Formats - Sealed Fate or Setting Out for New Shores? In: *Proc. of the IEEE Int. Conf. on Multimedia Computing and Systems; Florence, Italy* (1999), pp. 604–610. Republished in Multimedia - Tools and Applications, 11(3):267-279, 2000.

[BKW03] BOLL, S.; KRÖSCHE, J.; WEGENER, C.: Paper chase revisited - a Real World Game meets Hypermedia. In: *Proc. of the 14th Int. Conf. on Hypertext; Nottingham, UK*, ACM Press, Aug. 2003, pp. 126–127

[BM95] BOTAFOGO, R. A.; MOSSÉ, D.: The MORENA Model for Hypermedia Authoring and Browsing. In: *Proc. of the Int. Conf. on Multimedia Computing and Systems; Washington, DC, USA*, IEEE Computer Society Press, 1995, pp. 42–49

[BM02] BRUSILOVSKY, P.; MAYBURY, M. T.: From adaptive hypermedia to the adaptive Web. In: *Communications of the ACM* 45 (2002), no. 5, pp. 30–33

[BM03] BECKETT, D.; MCBRIDE, B.: RDF/XML Syntax Specification (Revised) - W3C Recommendation. Feb. 2003. Retrieved March 6, 2006, from http://www.w3.org/TR/rdf-syntax-grammar/

[BMM+99] BOSCH, J.; MOLIN, P.; MATTSSON, M.; BENGTSSON, P.; FAYAD, M. E.: Framework Problems and Experiences. In: FAYAD et al. [FSJ99a], pp. 55–82

[BMR+96] BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M.: *A System of Patterns (Pattern-Oriented Software Architecture, vol. 1)*. Wiley Series in Software Design Patterns, John Wiley & Sons, Jul. 1996

[Bol01] BOLL, S.: *ZYX - Towards flexible multimedia document models for reuse and adaptation*. PhD thesis, Vienna University, Austria, 2001

[Bol02] BOLL, S.: Modular Content Personalization Service Architecture for E-Commerce Applications. In: *Proc. of the 4th IEEE Int. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems; Newport Beach, CA, USA*, IEEE Computer Society Press, 2002, pp. 213–220

[Bol03a] BOLL, S.: MM4U - A framework for creating personalized multimedia content. In: *Proc. Int. Conf. on Distributed Multimedia Systems*, Sept. 2003, pp. 12–16

[Bol03b] BOLL, S.: Vienna 4 U - What Web Services can do for personalized multimedia applications. In: *7th Multi-Conf. on Systemics, Cybernetics and Informatics; Orlando, FL, USA*, Jul. 2003, pp. 220–225

[Bol05] BOLL, S.: Personalisierungsstrategien im Netz (Personalizationstrategies in the Net). In: TAEGER, J.; WIEBE, A. (ed.), *Mobilität Telematik Recht: Informationstechnik und Recht*, Köln, Germany: Dr. Otto Schmidt, vol. 14, 2005, pp. 77–93

[Bos00] BOSCH, J.: *Design and Use of Software Architectures: Adopting and evolving a product-line approach*. Addison-Wesley, 2000

[BRHO99] BULTERMAN, D. C. A.; RUTLEDGE, L.; HARDMAN, L.; VAN OSSENBRUGGEN, J.: Supporting Adaptive and Adaptable Hypermedia Presentation Semantics. In: *The 8th IFIP 2.6 Working Conf. on Database Semantics: Semantic Issues in Multimedia Systems; Rotorua, New Zealand*, Jan. 1999

[BRL91] BULTERMAN, D. C. A.; VAN ROSSUM, G.; VAN LIERE, R.: A Structure of Transportable, Dynamic Multimedia Documents. In: *Proc. of the Summer 1991 Usenix Conf.; Nashville, TN, USA*, 1991

[Bru94] BRUSILOVSKY, P.: Adaptive Hypermedia: An Attempt to Analyze and Generalize. In: BRUSILOVSKY, P. A. M., P. KOMMERS; STREITZ, N. A. (ed.), *First Int. Conf. Multimedia, Hypermedia, and Virtual Reality: Models, Systems, and Applications; Moscow, Russia*, Springer-Verlag, Sept. 1994, vol. 1077 of *Lecture Notes in Computer Science*, pp. 288–304

[Bru96] BRUSILOVSKY, P.: Methods and techniques of adaptive hypermedia. In: *User Modeling and User-Adapted Interaction* 6 (1996), no. 2-3, pp. 87–129

[BSZM02] BRAILSFORD, T. J.; STEWART, C. D.; ZAKARIA, M. R.; MOORE, A.: Autonavigation, Links and Narrative in an Adaptive Web-Based Integrated Learning Environment. In: *World Wide Web Conf.;*

Honolulu, HI, USA, May 2002. Retrieved March 7, 2006, from http://www2002.org/CDROM/alternate/738/

[Bul95] BULTERMAN, D. C. A.: Embedded video in hypermedia documents: supporting integration and adaptive control. In: *ACM Trans. Inf. Syst.* 13 (1995), no. 4, pp. 440–470

[BV03] BESZTERI, I.; VUORIMAA, P.: Layout Adaptation with XFrames. In: *The 9th Int. Conf. on Distributed Multimedia Systems; Miami, FL, USA*, Sept. 2003, pp. 199–203

[BW03] BOLL, S.; WESTERMANN, U.: MediÆther - an Event Space for Context-Aware Multimedia Experiences. In: *Proc. of the Int. ACM SIGMM Workshop on Experiential Telepresence; Berkeley, CA, USA*, Nov. 2003, pp. 21–30

[BWH+03] BURNETT, I.; VAN DE WALLE, R.; HILL, K.; BORMANS, J.; PEREIRA, F.: MPEG-21: Goals and Achievements. In: *IEEE MultiMedia* 10 (2003), no. 4, pp. 60–70

[BZST04] BINEMANN-ZDANOWICZ, A.; SCHEWE, K.-D.; THALHEIM, B.: Adaptation to Learning Styles. In: KINSHUK; LOOI, C.-K.; SUTINEN, E.; SAMPSON, D. G.; AEDO, I.; UDEN, L.; KÄHKÖNEN, E. (ed.), *Proc. of the IEEE Int. Conf. on Advanced Learning Technologies; Joensuu, Finland*, IEEE Computer Society Press, Aug./Sept. 2004, pp. 121–125

[CBB+03] CLEMENTS, P.; BACHMANN, F.; BASS, L.; GARLAN, D.; IVERS, J.; LITTLE, R.; NORD, R.; STAFFORD, J.: *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2003

[CC02] CAREY, J.; CARLSON, B.: *Framework Process Patterns: Lessons Learned Developing Application Frameworks*. Addison-Wesley, 2002

[CCI93] CCITT (INT. TELEGRAPH AND TELEPHONE CONSULTATIVE COMMITTEE): Terminal Equipment and Protocols for Telematic Services: Information Technology, Digital Compression and Coding of Continuous-tone Still Images, Requirements and Guidelines - Recommendation T.81. 1993. Retrieved February 7, 2006, from http://www.w3.org/Graphics/JPEG/itu-t81.pdf

[CD01] CHEESMAN, J.; DANIELS, J.: *UML Components: A Simple Process for Specifying Component-Based Software*. Addison-Wesley, 2001

[CFMP99] CERI, S.; FRATERNALI, P.; MAURINO, A.; PARABOSCHI, S.: One-to-One Personalization of Data-Intensive Web Sites. In: *WebDB (Informal Proceedings)*, 1999, pp. 1–6

[CGL+02] CONSOLE, L.; GIORIA, S.; LOMBARDI, I.; SURANO, V.; TORRE, I.: Adaptation and Personalization on Board Cars: A Framework and Its Application to Tourist Services. In: *Proc. of the Second Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems; London, UK*, Springer-Verlag, 2002, pp. 112–121

[Chi89] CHIN, D. N.: KNOME: Modeling What the User Knows in UC. In: KOBSA and WAHLSTER [KW89], pp. 74–107

[CJC02] CHRISTIANSSON, B.; JAKOBSSON, L.; CRNKOVIC, I.: CBD Process. In: CRNKOVIC and LARSSON [CL02], ch. 5, pp. 89–113

[CK00] CHEN, G.; KOTZ, D.: *A Survey of Context-Aware Mobile Computing Research*. tech. rep. TR2000-381, Dartmouth College, Department of Computer Science, Hanover, NH, USA, Nov. 2000

[CKAH02] CHEONG, S. N.; KAM, H. S.; AZHAR, K. M.; HANMANDLU, M.: Course Authoring and Management System for Interactive and Personalized Multimedia Online Notes through XSL, XSLT, and SMIL. In: *Interactive Computer Aided Learning; Villach, Austria*, Sept. 2002

[CL02] CRNKOVIC, I.; LARSSON, M. (ed.): *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002

[CLM02] CRANOR, L.; LANGHEINRICH, M.; MARCHIORI, M.: A P3P Preference Exchange Language 1.0 (APPEL1.0): W3C Working Draft. Apr. 2002. Retrieved May 22, 2006, from http://www.w3.org/TR/P3P-preferences/

[CMD02] CHEVERST, K.; MITCHELL, K.; DAVIES, N.: The role of adaptive hypermedia in a context-aware tourist GUIDE. In: *Communications of the ACM* 45 (2002), no. 5, pp. 47–51

[Coc06] COCKBURN, A.: Crystal Main Foyer. 2006. Retrieved March 6, 2006, from http://alistair.cockburn.us/crystal/crystal.html

[Com90] COMPUSERVE INC., USA: Graphics Interchange Format (GIF) Version 89a. Jul. 1990. Retrieved February 7, 2006, from http://www.w3.org/Graphics/GIF/spec-gif89a.txt

[Com05] COMPUTER GRAPHICS LAB, SAARLAND UNIVERSITY, SAARBRÜCKEN, GERMANY: Network-Integrated Multimedia Middleware. 2005. Retrieved March 24, 2006, from http://www.networkmultimedia.org/

[Com06] COMMUNICATION AND MULTIMEDIA GROUP, TU BRAUNSCHWEIG, GERMANY: Multimedia Gateway Architecture for Adaptive Content Distribution. 2006. Retrieved March 6, 2006, from http://www.ibr.cs.tu-bs.de/projects/mmgw/

[Con00] CONLAN, O.: Novel components for supporting adaptivity in education systems - model-based integration approach. In: *Proc. of the 8th ACM Int. Conf. on Multimedia; Marina del Rey, CA, USA*, ACM Press, 2000, pp. 519–520

[Cou90] COUSOT, P.: Methods and Logics for Proving Programs. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, Elsevier, 1990, pp. 841–994

[CP01] CANNATARO, M.; PUGLIESE, A.: A Flexible Architecture for Adaptive Hypermedia Systems. In: *Int. Workshop on Intelligent Techniques for Web Personalization; Seattle, WA, USA*, 2001

[CSI05] CSIRO, AUSTRALIA: PocketSVG. 2005. Retrieved November 2, 2005, from http://www.pocketsvg.com/

[CSI06] CSIRO, AUSTRALIA: Welcome to Annodex.net: Open standards for annotating and indexing networked media. 2006. Retrieved February 1, 2006, from http://www.annodex.net/

[CTB+05] CHRISTODOULAKIS, S.; TSINARAKI, C.; BREITENEDER, C.; EIDENBERGER, H.; DIVOTKEY, D.; BOLL, S.; SCHERP, A.; BERTINO, E.; PEREGO, A.: CoCoMA: Content and Context Aware Multimedia Content Retrieval, Delivery and Presentation. In: *Proc. of European*

Conf. on Research and Advanced Technology for Digital Libraries - DELOS posters; Vienna, Austria, 2005, p. 33 ff.

[CWI05] CWI, THE NETHERLANDS: The AMBULANT Project. 2005. Retrieved March 6, 2006, from http://www.cwi.nl/projects/Ambulant

[DA99] DEY, A. K.; ABOWD, G. D.: *Towards a Better Understanding of Context and Context-Awareness*. tech. rep. GIT-GVU-99-22, Graphics, Visualization and Usability Center and College of Computing, Georgia Institute of Technology, Atlanta, GA, USA, Jun. 1999

[DB02] DE BRA, P.: Adaptive educational hypermedia on the web. In: *Communications of the ACM* 45 (2002), no. 5, pp. 60–61

[DBAB+03] DE BRA, P.; AERTS, A.; BERDEN, B.; DE LANGE, B.; ROUSSEAU, B.; SANTIC, T.; SMITS, D.; STASH, N.: AHA! The adaptive hypermedia architecture. In: *Proc. of the 14th ACM Conf. on Hypertext and hypermedia; Nottingham, UK*, Aug. 2003, pp. 81–84

[DBAHW00] DE BRA, P.; AERTS, A.; HOUBEN, G.-J.; WU, H.: Making General-Purpose Adaptive Hypermedia Work. In: *Proc. of WebNet 2000 - World Conf. on the WWW and Internet; San Antonio, TX, USA*, Association for the Advancement of Computing in Education, VA, USA, 2000, pp. 117–123

[DBASS02] DE BRA, P.; AERTS, A.; SMITS, D.; STASH, N.: AHA! Version 2.0: More Adaptation Flexibility for Authors. In: *Proc. of the AACE ELearn'2002 Conf.*, Association for the Advancement of Computing in Education, VA, USA, Oct. 2002, pp. 240–246

[DBBC02] DE BRA, P. D.; BRUSILOVSKY, P.; CONEJO, R. (ed.): *Proc. of the Second Int. Conf. for Adaptive Hypermedia and Adaptive Web-Based Systems; Malaga, Spain*, Springer-Verlag, May 2002

[DBBH99] DE BRA, P.; BRUSILOVSKY, P.; HOUBEN, G.-J.: Adaptive hypermedia: from systems to framework. In: *ACM Computing Surveys* 31 (1999), no. 4, p. 12

[DBHW99] DE BRA, P.; HOUBEN, G.-J.; WU, H.: AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In: *Proc. of the 10th ACM Conf. on Hypertext and hypermedia: returning to our diverse roots; Darmstadt, Germany*, ACM Press, 1999, pp. 147–156

[DCMF99] DAVIES, N.; CHEVERST, K.; MITCHELL, K.; FRIDAY, A.: Caches in the Air: Disseminating Information in the Guide System. In: *2nd IEEE Workshop on Mobile Comp. Systems and App.; New Orleans, USA*, 1999, pp. 11–19

[DCRA+98] DE CAROLIS, B.; DE ROSIS, F.; ANDREOLI, C.; CAVALLO, V.; DE CICCO, M. L.: The Dynamic Generation of Hypertext Presentations of Medical Guidelines. In: *The New Review of Hypermedia and Multimedia* 4 (1998), pp. 67–88

[DCRBM99] DE CAROLIS, B.; DE ROSIS, F.; BERRY, D.; MICHAS, I.: Evaluating plan-based hypermedia generation. In: *Proc. of European Workshop on Natural Language Generation; Toulouse, France*, 1999, pp. 126–134

[DDV03] DOULAMIS, N. D.; DOULAMIS, A. D.; VARVARIGOU, T. A.: Adaptive Algorithms for Interactive Multimedia. In: *IEEE MultiMedia* 10. (2003), no. 4, pp. 38–47

[DE05] DIVOTKEY, D.; EIDENBERGER, H.: Content-based Querying Embedded in Multimedia Presentations. In: *IEEE Multimedia Signal Processing Workshop; Shanghai, China*, 2005

[DEL06] DELOS NOE: DELOS Network of Excellence on Digital Libraries. 2006. Retrieved March 6, 2006, from http://www.delos.info/

[DFG+04] DZIARSTEK, C.; FARNSCHLÄDER, F.; GILLESSEN, S.; SÜSSMILCH-WALTHER, I.; WINKLER, V.: A User-Aware Financial Advisory System. In: CHAMONI, P.; DEITERS, W.; N., G.; ET AL. (ed.), *Multikonferenz Wirtschaftsinformatik; University of Duisburg/Essen, Germany*, Aka GmbH, Berlin, Germany, 2004, pp. 217–229

[DFK06] DFKI GMBH, GERMANY: SmartKom: Dialog-based Human-Technology Interaction by Coordinated Analysis and Generation of Multiple Modalities. Mar. 2006. Retrieved March 28, 2006, from http://www.smartkom.org/start_en.html

[DFM+96] DALAL, M.; FEINER, S.; MCKEOWN, K.; PAN, S.; ZHOU, M.; HÖLLERER, T.; SHAW, J.; FENG, Y.; FROMER, J.: Negotiation for automated generation of temporal multimedia presentations. In: *Proc. of the 4th ACM Int. Conf. on Multimedia; Boston, MA, USA*, ACM Press, 1996, pp. 55–64

[DH98] DAWSON, F.; HOWES, T.: vCard MIME Directory Profile (RFC 2426). Sept. 1998. Retrieved April 4, 2006, from http://www.ietf.org/rfc/rfc2426.txt

[DHM02] DACHSELT, R.; HINZ, M.; MEISSNER, K.: Contigra: an XML-based architecture for component-oriented 3D applications. In: *Proc. of the 7th Int. Conf. on 3D Web technology; Tempe, AZ, USA*, ACM Press, Feb. 2002, pp. 155–163

[Dig90] DIGMAN, J. M.: Personality structure: Emergence of the five factor model. In: *Annual Review of Psychology* 41 (1990), pp. 417–440

[DK95] DUDA, A.; KERAMANE, C.: Structured Temporal Composition of Multimedia Data; Blue Mountain Lake, NY, USA. In: *Proc. of the IEEE Int. Workshop Multimedia-Database-Management Systems*, IEEE Computer Society Press, 1995, pp. 136–142

[DT03] DOGRU, A. H.; TANIK, M. M.: A Process Model for Component-Oriented Software Engineering. In: *IEEE Software* 20 (2003), no. 2, pp. 34–41

[Dub02] DUBLIN CORE METADATA INITIATIVE: Expressing Simple Dublin Core in RDF/XML. Jul. 2002. Retrieved March 6, 2006 from http://dublincore.org/documents/2002/07/31/dcmes-xml/

[DW99] D'SOUZA, D. F.; WILLS, A. C.: *Objects, Components, and Frameworks with UML: The Catalysis Approach*. Addison-Wesley, 1999

[EB90] VAN EMDE BOAS, P.: Machine Models and Simulation. In: *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, Elsevier, 1990, pp. 1–66

[EB03] EIDENBERGER, H.; BREITENEDER, C.: VizIR - A framework for visual information retrieval. In: *Journal of Visual Languages and Computing* 14 (2003), no. 5, pp. 443–469

[Ecl06] ECLIPSE FOUNDATION, CANADA: Eclipse.org home. 2006. Retrieved March 6, 2006, from http://www.eclipse.org/

[EF91] EGENHOFER, M. J.; FRANZOSA, R.: Point-Set Topological Spatial Relations. In: *Int. Journal of Geographic Information Systems* 5 (1991), no. 2, pp. 161–174

[EFMS91] ELHADAD, M.; FEINER, S.; MCKEOWN, K.; SELIGMANN, D.: Generating customized text and graphics in the COMET explanation testbed. In: *Proc. of the 23rd conf. on Winter simulation; Phoenix, AZ, USA*, IEEE Computer Society Press, 1991, pp. 1058–1065

[Eid03] EIDENBERGER, H.: Media handling for visual information retrieval in VizIR. In: *Visual Communications and Image Processing* 5150 (2003), pp. 1078–1088

[EKMT03] EGGES, A.; KSHIRSAGAR, S.; MAGNENAT-THALMANN, N.: A Model for Personality and Emotion Simulation. In: PALADE, V.; HOWLETT, R. J.; JAIN, L. C. (ed.), *Knowledge-Based Intelligent Information & Engineering Systems*, Springer-Verlag, Sept. 2003, vol. 2773 of *Lecture Notes in Computer Science*, pp. 453–461

[EKMT04] EGGES, A.; KSHIRSAGAR, S.; MAGNENAT-THALMANN, N.: Generic personality and emotion simulation for conversational agents. In: *Journal of Visualization and Computer Animation* 15 (2004), no. 1, pp. 1–13

[ELVKB01] VAN DEN EIJKEL, G. C.; LANKHORST, M. M.; VAN KRANENBURG, H.; BIJLSMA, M. E.: Personalization of next-generation mobile services. In: *Wireless World Research forum; Munich, Germany*, Mar. 2001

[EN03] ELMASRI, R.; NAVATHE, S.: *Fundamentals of database systems*. Addison-Wesley, 4th edition, 2003

[Enc05] ENCYCLOPAEDIA BRITANNICA, INC., CHICAGO, USA: Encyclopaedia Britannica 2005 Ultimate Reference Suite DVD. 2005

[ER03] ENDRES, A.; ROMBACH, D.: *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories*. Pearson and Addison-Wesley, 2003

[ESN03] ENGELS, G.; SAUER, S.; NEU, B.: Integrating Software Engineering and User-centred Design for Multimedia Software Developments. In: *Proc. IEEE Symposia on Human-Centric Computing Languages and Environments : Symposium on Visual / Multimedia Software Engineering; Auckland, New Zealand*, IEEE Computer Society Press, Oct. 2003, pp. 254–256

[Exo05] EXOR INT. INC., USA: eSVG: Embedded SVG. 2005. Retrieved March 6, 2006, from http://esvg.ultimodule.com/bin/esvg/templates/

[FHBP03] FRASINCAR, F.; HOUBEN, G.-J.; BARNA, P.; PAU, C.: RDF/XML-based Automatic Generation of Adaptable Hypermedia Presentations. In: *Int. Symposium on Information Technology; Las Vegas, NV, USA*, IEEE Computer Society Press, Apr. 2003, pp. 410–414

[FHLS99] FROEHLICH, G.; HOOVER, H. J.; LIU, L.; SORENSON, P.: Reusing Hooks. In: FAYAD et al. [FSJ99a], pp. 219–236

[FHV02] FRASINCAR, F.; HOUBEN, G.; VDOVJAK, R.: Specification Framework for Engineering Adaptive Web Applications. In: *Proc. of the 11th Int. World Wide Web Conf.; Honolulu, HI, USA*, May 2002. Retrieved August 03, 2006, from http://www2002.org/CDROM/alternate/682/

[FIP02] FIPA (FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS), SWITZERLAND: FIPA Device Ontology Specification - Version E. Dec. 2002. Retrieved March 6, 2006, from http://www.fipa.org/specs/fipa00091/

[Fis01] FISCHER, G.: User Modeling in Human-Computer Interaction. In: *User Modeling and User-Adapted Interaction* 11 (2001), no. 1-2, pp. 65–86

[FK02] FINK, J.; KOBSA, A.: User Modeling for Personalized City Tours. In: *Artificial Intelligence Review* 18 (2002), no. 1, pp. 33–74

[FKNS02] FINK, J.; KOENEMANN, J.; NOLLER, S.; SCHWAB, I.: Putting personalization into practice. In: *Communications of the ACM* 45 (2002), no. 5, pp. 41–42

[FKS97a] FINK, J.; KOBSA, A.; SCHRECK, J.: Personalized Hypermedia Information Provision Through Adaptive and Adaptable System Features: User Modelling, Privacy and Security Issues. In: *Proc. of the 4th Int. Conf. on Intelligence and Services in Networks; London, UK*, Springer-Verlag, 1997, pp. 459–467

[FKS97b] FINK, J.; KOBSA, A.; SCHRECK, J.: Personalized Hypermedia Information through Adaptive and Adaptable System Features: User Modeling, Privacy and Security Issues. In: MULLERY, A.; BESSON, M.; CAMPOLARGO, M.; GOBBI, R.; REED, R. (ed.), *Intelligence in Services and Networks: Technology for Cooperative Competition*, Springer-Verlag, 1997, pp. 459–467

[FM93] FEINER, S. K.; MCKEOWN, K. R.: Automating the generation of coordinated multimedia explanations. In: MAYBURY [May93], ch. 5, pp. 117–138

[FPR00] FONTOURA, M.; PREE, W.; RUMPE, B.: UML-F: A Modeling Language for Object-Oriented Frameworks. In: BERTINO, E. (ed.), *European Conf. on Object-Oriented Programing*, Springer-Verlag, 2000, vol. 1850 of *Lecture Notes in Computer Science*, pp. 63–82

[FPR02] FONTOURA, M.; PREE, W.; RUMPE, B.: *The UML Profile for Framework Architectures*. Object Technology Series, Addison-Wesley, 2002

[FR96] FACONTI, G. P.; RIST, T. (ed.): *Proc. of ECAI-96 Workshop Towards a Standard Reference Model for Intelligent Multimedia Presentation Systems; Budapest, Hungary*, 1996

[Fra06] FRAUNHOFER IIS, GERMANY: MPEG Audio Layer-3. 2006. Retrieved February 7, 2006, from http://www.iis.fraunhofer.de/amm/techinf/layer3/

[FRF+02] FOWLER, M.; RICE, D.; FOEMMEL, M.; HIEATT, E.; MEE, R.; STAFFORD, R. (ed.): *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002

[Fri06] FRIEDRICH-ALEXANDER UNIVERSITY OF ERLANGEN-NUREMBERG, GERMANY: The RETAVIC project. 2006. Retrieved March 24, 2006, from http://www6.informatik.uni-erlangen.de/research/projects/retavic/

[FSJ99a] FAYAD, M. E.; SCHMIDT, D. C.; JOHNSON, R. E. (ed.): *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. John Wiley & Sons, 1999

[FSJ99b] FAYAD, M. E.; SCHMIDT, D. C.; JOHNSON, R. E. (ed.): *Implementing Application Frameworks: Object-Oriented Frameworks at Work*. Wiley computer publishing, John Wiley & Sons, 1999

[Fuh94] FUHRT, B.: Multimedia Systems: An Overview. In: *IEEE MultiMedia* 1 (1994), no. 1, pp. 47–59

[GAO95] GARLAN, D.; ALLEN, R.; OCKERBLOOM, J.: Architectural mismatch or why it's hard to build systems out of existing parts. In: *Proc. of the 17th Int. Conf. on Software engineering; Seattle, WA, USA*, ACM Press, 1995, pp. 179–185

[GBE96] GÖTZE, R.; BOLES, D.; EIRUND, H.: Multimedia User Interfaces. In: SONNENSCHEIN, M. (ed.), *Final Report Arbeitsgruppe Informatik-Systeme*, Carl von Ossietzky University, Oldenburg, Germany, ch. 6, Nov. 1996, pp. 120–165

[GBS01] VAN GURP, J.; BOSCH, J.; SVAHNBERG, M.: On the Notion of Variability in Software Product Lines. In: KAZMAN, R.; KRUCHTEN, P.; VERHOEF, C.; VAN VLIET, H. (ed.), *Proc.of the Working IEEE/IFIP Conf. on Software Architecture; Amsterdam, The Netherlands*, IEEE Computer Society Press, 2001, pp. 45–54

[GC96] GOLOSHUBIN, A.; COOK, J.: JavaBrowser - display an HTML file from within an application or an applet. Dec. 1996. Retrieved April 20, 2006, from http://www.ii.uib.no/~alexey/jb/index.html

[Geu02] GEURTS, J.: *Constraints for Multimedia Presentation Generation*. Master's thesis, University of Amsterdam, Amsterdam, The Netherlands, Jan. 2002

[GHJV04] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J.: *Design patterns: elements of reusable object-oriented software*. Programmer's Choice, Addison-Wesley, Jul. 2004

[Gir06] GIROW, A.: TinyLine SVG. 2006. Retrieved March 6, 2006, from http://www.tinyline.com/svgt/index.html

[GL03] GEIGEL, J.; LOUI, A. C.: Using Genetic Algorithms for Album Page Layouts. In: *IEEE MultiMedia* 10 (2003), no. 4, pp. 16–27

[Gla04] GLASS, R. L.: *Facts and Fallacies of Software Engineering*. Addison-Wesley, 4th edition, 2004

[GLMR01] GROSSMANN, M.; LEONHARDI, A.; MITSCHANG, B.; ROTHERMEL, K.: A World Model for Location-Aware Systems. In: *Informatik* 8 (2001), no. 5, pp. 22–25

[GOH01] GEURTS, J.; VAN OSSENBRUGGEN, J.; HARDMAN, L.: Application-specific constraints for Multimedia Presentation Generation. In: *Proc. of the Int. Conf. on Multimedia Modeling; Amsterdam, The Netherlands*, IEEE Computer Society Press, Nov. 2001, pp. 247–266

[Gov99] GOVONI, D.: *Java Application Frameworks*. John Wiley & Sons, 1999

[GR98] GREINER, C.; ROSE, T.: A Web Based Training System for Cardiac Surgery: The Role of Knowledge Management for Interlinking Information Items. In: *Proc. The World Congress on the Internet in Medicine; London, UK*, Nov. 1998

[Gra95] GRAF, W. H.: The constraint-based layout framework LayLab and its applications. In: *Proc. of ACM Workshop on Effective Abstractions in Multimedia, Layout and Interaction; San Francisco, CA, USA*, 1995

[Gra97] GRAF, W. H.: Intelligent multimedia layout: a reference architecture for the constraint-based spatial layout of multimedia presentations. In: *Comput. Stand. Interfaces* 18 (1997), no. 6-7, pp. 515–524

[GS01] GROSS, T.; SPECHT, M.: Awareness in Context-Aware Information Systems. In: *Mensch und Computer; Bad Honneff (Bonn), Germany*, 2001, pp. 173–182

[GT95] GIBBS, S. J.; TSICHRITZIS, D. C.: *Multimedia Programming: Objects, Environments and Frameworks*. ACM Press Books, Addison-Wesley, 2nd edition, 1995

[Har98] HARDMAN, L.: *Modeling and Authoring Hypermedia Documents*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, Mar. 1998

[Har05] HARDMAN, L.: Canonical processes of media production. In: *Proc. of the ACM workshop on Multimedia for human communication; Hilton, Singapore*, ACM Press, 2005, pp. 1–6

[Has02] HASSELBRING, W.: Component-Based Software Engineering. In: CHANG, S. K. (ed.), *Handbook of Software Engineering and Knowledge Engineering*, World Scientific, 2002, pp. 289–305

[HBR94] HARDMAN, L.; BULTERMAN, D. C. A.; VAN ROSSUM, G.: The Amsterdam hypermedia model: adding time and context to the Dexter model. In: *Communications of the ACM* 37 (1994), no. 2

[HCV04] HONKALA, M.; CESAR, P.; VUORIMAA, P.: A Device Independent XML User Agent for Multimedia Terminals. In: *Proc. of the IEEE 6th Int. Symposium on Multimedia Software Engineering; Miami, FL, USA*, IEEE Computer Society Press, Dec. 2004, pp. 116–123

[HDM03] HOFFMANN, H.; DACHSELT, R.; MEISSNER, K.: An Independent Declarative 3D Audio Format on the Basis of XML. In: *Proc. of the 2003 Int. Conf. on Auditory Display; Boston, MA, USA*, Boston University Publications Production Department, Jul. 2003, pp. 99–102

[HFK95] HIRZALLA, N.; FALCHUK, B.; KARMOUCH, A.: A Temporal Model for Interactive Multimedia Scenarios. In: *IEEE Multimedia* 2 (1995), no. 3, pp. 24–31

[HHMW06] HÄRDER, T.; HAUSTEIN, M.; MATHIS, C.; WAGNER, M.: Node labeling schemes for dynamic XML documents reconsidered. In: *Data & Knowledge Engineering* 58 (2006)

[HLZ04] HUA, X.-S.; LU, L.; ZHANG, H.-J.: P-Karaoke: personalized karaoke system. In: *Proc. of the 12th annual ACM Int. Conf. on Multimedia; New York, NY, USA*, ACM Press, 2004, pp. 172–173

[HMHGK01] HELLER, R. S.; MARTIN, C. D.; HANEEF, N.; GIEVSKA-KRLIU, S.: Using a theoretical multimedia taxonomy framework. In: *J. Educ. Resour. Comput.* 1 (2001), no. 1es, p. 6

[Hod02] HODGES, M.: Is Web Business Good Business. Mar. 2002. Retrieved February 28, 2006, from http://www.technologyreview.com/InfoTech/wtr_11578,258,p1.html

[HRJM94] HARDMAN, L.; VAN ROSSUM, G.; JANSEN, J.; MULLENDER, S.: CMIFed: a transportable hypermedia authoring system. In: *Proc. of the 2nd ACM Int. Conf. on Multimedia; San Francisco, CA, USA*, 1994, pp. 471–472

[HS91] HERRTWICH, R. G.; STEINMETZ, R.: Towards Integrated Multimedia Systems: Why and How. In: ENCARNAÇÃO, J. L. (ed.), *Jahrestagung Gesellschaft für Informatik*, Springer-Verlag, 1991, vol. 293 of *Informatik-Fachberichte*, pp. 327–342

[Hun99] HUNTER, J.: Multimedia Metadata Schemas. Oct. 1999. Retrieved March 6, 2006, from http://www2.lib.unb.ca/Imaging_docs/IC/schemas.html

[HVL01] HJELSVOLD, R.; VDAYGIRI, S.; LÉAUTÉ, Y.: Web-based personalization and management of interactive video. In: *World Wide Web Conf.*, 2001, pp. 129–139

[IB05] IGNATOVA, T.; BRUDER, I.: Utilizing a Multimedia UML Framework for an Image Database Application. In: AKOKA, J.; LIDDLE, S. W.; SONG, I.-Y.; BERTOLOTTO, M.; COMYN-WATTIAU, I.; CHERFI, S. S.-S.; VAN DEN HEUVEL, W.-J.; THALHEIM, B.; KOLP, M.; BRESCIANI, P.; TRUJILLO, J.; KOP, C.; MAYR, H. C. (ed.), *ER (Workshops); Klagenfurt, Austria*, Springer-Verlag, 2005, vol. 3770 of *Lecture Notes in Computer Science*, pp. 23–32

[IBM04a] IBM CORPORATION, USA: IBM Research - Video Semantic Summarization Systems. 2004. Retrieved March 6, 2006, from http://www.research.ibm.com/MediaStar/VideoSystem.html

[IBM04b] IBM CORPORATION, USA: QBIC Home Page. 2004. Retrieved March 6, 2006, from http://wwwqbic.almaden.ibm.com/

[IBM06a] IBM CORPORATION, USA: IBM MPEG-4 Technologies. Apr. 2006. Retrieved February 8, 2006, from http://www.research.ibm.com/mpeg4/

[IBM06b] IBM CORPORATION, USA: IBM Toolkit for MPEG-4. Apr. 2006. Retrieved May 12, 2006, from http://www.alphaworks.ibm.com/tech/tk4mpeg4/

[IEE00] IEEE (INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS): *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems - Description*. Standard IEEE 1471-2000, ANSI/IEEE, 2000

[INR02] INRIA RHÔNE-ALPES, FRANCE: PocketSMIL 2.0. 2002. Retrieved March 6, 2006, from http://wam.inrialpes.fr/software/pocketsmil/

[ISO96] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC 1/SC 34: Document Style Semantics and Specification Language (DSSSL), ISO/IEC 10179:1996. 1996. Geneva, Switzerland

[ISO97] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC 1/SC 34: Information technology - Hypermedia/Time-based Structuring Language (HyTime), ISO/IEC 10744:1997. 1997. Geneva, Switzerland

[ISO99] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: MPEG-7: Context, Objectives and Technical Roadmap, V.12. ISO/IEC Document N2861. Jul. 1999. Geneva, Switzerland

[ISO01a] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION): ISO/IEC 14496-1:2001 Information technology - Coding of audio-visual objects - Part 1: Systems; Amendment 3 XMT. 2001. Geneva, Switzerland

[ISO01b] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: Information Technology - Multimedia Content Description Interface - Part 2: Description Definition Language. ISO/IEC Final Draft Int. Standard 15938-2:2001. Sept. 2001. Geneva, Switzerland

[ISO01c] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: Information Technology - Multimedia Content Description Interface - Part 3: Visual. ISO/IEC Final Draft Int. Standard 15938-3:2001. Jun. 2001. Geneva, Switzerland

[ISO01d] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: Information Technology - Multimedia Content Description Interface - Part 4: Audio. ISO/IEC Final Draft Int. Standard 15938-4:2001. Jul. 2001. Geneva, Switzerland

[ISO01e] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: Information Technology - Multimedia Content Description Interface - Part 5: Multimedia Description Schemes. ISO/IEC Final Draft Int. Standard 15938-5:2001. Oct. 2001. Geneva, Switzerland

[ISO01f] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: InformationTechnology - Multimedia Content Description Interface - Part 1: Systems. ISO/IEC Final Draft Int. Standard 15938-1:2001. Nov. 2001. Geneva, Switzerland

[ISO05] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION) JTC1/SC29/WG11: LASeR and SAF editor's study: ISO/IEC 14496-20 Study; Poznan, Poland. Jul. 2005. Geneva, Switzerland. Retrieved May 22, 2006, from http://www.mpeg-laser.org/documents/LASeR_specification_july2005.pdf

[ISO06] ISO/IEC (INT. ORGANIZATION FOR STANDARDIZATION/INT. ELECTROTECHNICAL COMMISSION): Information technology - Coding of audio-visual objects - Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF). Jun. 2006. Geneva, Switzerland. Retrieved July 12, 2006,

from http://standards.iso.org/ittf/PubliclyAvailableStandards/c041650_ISO_IEC_14496-20_2006(E).zip

[Jam98] JAMESON, A.: Adapting to the User's Time and Working Memory Limitations: New Directions of Research. In: TIMM, U. J.; RÖSSEL, M. (ed.), *Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen (Adaptivity and User Modeling in Interactive Software Systems); Erlangen, Germany*, FORWISS, 1998

[Jam01] JAMESON, A.: Modeling Both the Context and the User. In: *Personal and Ubiquitous Computing* 5 (2001), no. 1, pp. 29–33

[JB01] JOURDAN, M.; BES, F.: A new step towards multimedia documents generation. In: *Int. Conf. on Media Futures; Florence, Italy*, May 2001, pp. 25–28

[Jen01] JENSEN, F. V.: *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001

[JF88] JOHNSON, R. E.; FOOTE, B.: Designing Reusable Classes. In: *Journal of Object-Oriented Programming* 1 (1988), no. 2, pp. 22–35

[JLR+98] JOURDAN, M.; LAYAÏDA, N.; ROISIN, C.; SABRY-ISMAÏL, L.; TARDIF, L.: Madeus, an Authoring Environment for Interactive Multimedia Documents. In: *Proc.of the 6th ACM Int. Conf. on Multimedia; Bristol, UK*, ACM Press, 1998, pp. 267–272

[JLR99] JOURDAN, M.; LAYAÏDA, N.; ROISIN, C.: Authoring Techniques for Temporal Scenarios of Multimedia Documents. In: *Handbook of Internet and multimedia systems and applications*, CRC press, IEEE press, ch. 8, 1999, pp. 179–200

[JLS+04] JACOBS, C.; LI, W.; SCHRIER, E.; BARGERON, D.; SALESIN, D.: Adaptive document layout. In: *Communications of the ACM* 47 (2004), no. 8, pp. 60–66

[JLSI97] JOURDAN, M.; LAYAÏDA, N.; SABRY-ISMAIL, L.: Authoring Environment for Interactive Multimedia Documents. In: *Proc. the 13th Int. Conf. on Computer Communications; Cannes, France*, Nov. 1997, pp. 19–26

[JLTH06] JANNACH, D.; LEOPOLD, K.; TIMMERER, C.; HELLWAGNER, H.: A knowledge-based framework for multimedia adaptation. In: *Applied Intelligence* 24 (2006), no. 2, pp. 109–125

[JN99] JACOBSON, E. E.; NOWACK, P.: Frameworks and Patterns: Architectural Abstractions. In: FAYAD et al. [FSJ99a], pp. 29–54

[JS02] JIANG, C.; STEENKISTE, P.: A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing. In: BORRIELLO, G.; HOLMQUIST, L. E. (ed.), *Proc. of the 4th Int. Conf. on Ubiquitous Computing; Göteborg, Sweden*, 2002, pp. 246–263

[KB05] KRÖSCHE, J.; BOLL, S.: The xPOI Concept. In: STRANG, T.; LINNHOFF-POPIEN, C. (ed.), *Int. Workshop on Location- and Context-Awareness; Oberpfaffenhofen, Germany*, Springer-Verlag, 2005, vol. 3479 of *Lecture Notes in Computer Science*, pp. 113–119

[KBB04] KRÖSCHE, J.; BOLL, S.; BALDZER, J.: MobiDENK - Mobile Multimedia in Monument Conservation. In: *IEEE MultiMedia* 11 (2004), no. 2, pp. 72–77

[KCH+90] KANG, K. C.; COHEN, S. G.; HESS, J. A.; NOVAK, W. E.; PETERSON, A. S.: *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. tech. rep. CMU/SEI-90-TR-21 ESD 90-TR-222, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA, Nov. 1990

[KG02] KIM, J.; GIL, Y.: Deriving Acquisition Principles from Tutoring Principles. In: CERRI, S. A.; GOUARDÈRES, G.; PARAGUAÇU, F. (ed.), *Proc.of the 6th Int. Conf. on Intelligent Tutoring Systems; Biarritz, France and San Sebastian, Spain*, Springer-Verlag, Jun. 2002, pp. 661–670

[KGF99] KLAS, W.; GREINER, C.; FRIEDL, R.: Cardio-OP: Gallery of Cardiac Surgery. In: *IEEE Int. Conf. on Multimedia Computing and Systems; Florence, Italy*, Jul. 1999, pp. 1092–1095

[Kje91] KJELLDAHL, L.: Introduction. In: KJELLDAHL, L. (ed.), *Multimedia: Systems, Interaction and Applications*, Springer-Verlag, ch. 1, Apr. 1991, pp. 3–5

[KKP01] KOBSA, A.; KOENEMANN, J.; POHL, W.: Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. In: *The Knowledge Engineering Review*, Cambridge University Press, 2001, vol. 16, pp. 111–155

[KKRB04] KLANTE, P.; KRÖSCHE, J.; RATT, D.; BOLL, S.: First-year Students' Paper Chase: a Mobile Location-Aware Multimedia Game. In: *Proc. of the 12th annual ACM Int. Conf. on Multimedia; New York, NY, USA*, Oct. 2004, pp. 934–935

[KLCR02] KEATES, S.; LANGDON, P.; CLARKSON, P. J.; ROBINSON, P.: User Models and User Physical Capability. In: *User Modeling and User-Adapted Interaction* 12 (2002), no. 2-3, pp. 139–169

[Kle06] KLEIN, B.: incops: INtroduction to COgnitive PSychology. 2006. Retrieved May 8, 2006, from http://apsymac33.uni-trier.de:8080/incops/

[KMT02] KSHIRSAGAR, S.; MAGNENAT-THALMANN, N.: Virtual humans personified. In: *Proc. of the 1st Int. joint Conf. on Autonomous agents and multiagent systems; Bologna, Italy*, ACM Press, 2002, pp. 356–357

[KNV00] KRAMER, J.; NORONHA, S.; VERGO, J.: A user-centered design approach to personalization. In: *Communications of the ACM* 43 (2000), no. 8, pp. 44–48

[Kob93a] KOBSA, A.: Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen (Adaptivity and user modeling in interactive software systems). In: HERZOG, O.; CHRISTALLER, T.; SCHÜTT, D. (ed.), *Grundlagen und Anwendungen der Künstlichen Intelligenz, 17. Fachtagung für Künstliche Intelligenz; Humboldt-Universitaty, Berlin, Germany*, Springer-Verlag, Sept. 1993, Informatik Aktuell, pp. 152–166

[Kob93b] KOBSA, A.: User Modeling: Recent Work, Prospects and Hazards. In: SCHNEIDER-HUFSCHMIDT, M.; KÜHME, T.; MALINOWSKI, U. (ed.), *Adaptive User Interfaces: Principles and Practise; Amsterdam, The Netherlands*, Elsevier, 1993, pp. 111–128

[Kob95] KOBSA, A.: Supporting User Interfaces for All Through User Modeling. In: *Proc. of the 6th Int. Conf. on Human-Computer Interaction; Yokohama, Japan*, 1995, pp. 155–157

[Kob96] KOBSA, A.: Benutzermodell (User model). In: STRUBE et al. [SBF+96], p. 63

[Kob01] KOBSA, A.: Generic User Modeling Systems. In: *User Modeling and User-Adapted Interaction* 11 (2001), no. 1-2, pp. 49–63

[Koc02] KOCH, M.: Global Identity Management to Boost Personalization. In: SCHUBERT, P.; LEIMSTOLL, U. (ed.), *Proc. Research Symposium on Emerging Electronic Markets; Basel, Switzerland*, Institute for Business Economics (IAB), University of Applied Sciences, Basel, Switzerland, Sept. 2002, pp. 137–147

[Koe94a] KOEGEL BUFORD, J. F.: Architectures and issues for distributed multimedia systems. In: KOEGEL BUFORD, J. F. (ed.), *Multimedia systems*, ACM Press, SIGGRAPH series, ch. 3, Dec. 1994, pp. 45–64

[Koe94b] KOEGEL BUFORD, J. F.: Uses of multimedia information. In: KOEGEL BUFORD, J. F. (ed.), *Multimedia systems*, ACM Press, SIGGRAPH series, ch. 1, Dec. 1994, pp. 1–25

[KPMM01] KORVA, J.; PLOMP, J.; MÄÄTTÄ, P.; METSO, M.: On-Line Service Adaptation for Mobile and Fixed Terminal Devices. In: *Proc. of the 2nd Int. Conf. on Mobile Data Management; London, UK*, Springer-Verlag, 2001, pp. 252–259

[KPW04] KING, R.; POPITSCH, N.; WESTERMANN, U.: METIS: a flexible database foundation for unified media management. In: *Proc.of the 12th annual ACM Int. Conf. on Multimedia*, ACM Press, 2004, pp. 744–745

[Krö01] KRÖNER, A.: *Adaptive Layout of Dynamic Web Pages*. Aka GmbH, Berlin, Germany, 2001. PhD thesis, Universität des Saarlandes, Germany

[Kru00] KRUCHTEN, P.: *The Rational Unified Process: an introduction*. Addison-Wesley Object Technology Series, Addison-Wesley, 2nd printing edition, 2000

[KRW+04] KLYNE, G.; REYNOLDS, F.; WOODROW, C.; OHTO, H.; HJELM, J.; BUTLER, M. H.; TRAN, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0 - W3C Recommendation. Jan. 2004. Retrieved March 6, 2006, from http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/

[KS04] KRÖNER, A.; SATO, H.: Personalizing the Appearance of Content Packages. In: ABECKER, A.; BICKEL, S.; BREFELD, U.; DROST, I.; HENZE, N.; HERDEN, O.; MINOR, M.; SCHEFFER, T.; STOJANOVIC, L.; WEIBELZAHL, S. (ed.), *Lernen - Wissensentdeckung - Adaptivität; Berlin, Germany*, Humbold-University Berlin, Germany, 2004, pp. 42–47

[Ksh02] KSHIRSAGAR, S.: A multilayer personality model. In: *Proc. of the 2nd Int. symposium on Smart graphics; Hawthorne, NY, USA*, ACM Press, 2002, pp. 107–115

[KW89] KOBSA, A.; WAHLSTER, W. (ed.): *User Models in Dialog Systems*. Springer-Verlag, 1989

[KW06] KIM, M.; WOOD, S.: XMT: MPEG-4 Textual Format for Cross-Standard Interoperability. 2006. Retrieved March 6, 2006, form http://www.research.ibm.com/mpeg4/Projects/XMTInterop.htm

[KWC00] KIM, M.; WOOD, S.; CHEOK, L.-T.: Extensible MPEG-4 Textual Format (XMT). In: *Proc. of the 8th ACM Multimedia Conf; Los Angeles, CA, USA*, Nov. 2000, pp. 71–74

[KYME03] KINNO, A.; YONEMOTO, Y.; MORIOKA, M.; ETOH, M.: Environment Adaptive XML Transformation and Its Application to Content Delivery. In: *Proc. of the 2003 Symposium on Applications and the Internet; Washington, DC, USA*, IEEE Computer Society Press, 2003, pp. 31–38

[Las06] LASZLO SYSTEMS, INC., CA, USA: OpenLaszlo. 2006. Retrieved February 7, 2006, from http://www.openlaszlo.org/

[LBBN04] LEHMANN, O.; BAUER, M.; BECKER, C.; NICKLAS, D.: From Home to World - Supporting Context-aware Applications through World Models. In: *Proc. of the 2nd IEEE Int. Conf. on Pervasive Computing and Communications; Orlando, FL, USA*, IEEE Computer Society Press, 2004, pp. 297–307

[LBHB99] LUNDBERG, L.; BOSCH, J.; HÄGGANDER, D.; BENGTSSON, P.-O.: Quality Attributes in Software Architecture Design. In: *Proc. of the IASTED 3rd Int. Conf. on Software Engineering and Applications; Scottsdale, AZ, USA*, Oct. 1999, pp. 353–362

[LG90a] LITTLE, T. D. C.; GHAFOOR, A.: Network considerations for distributed multimedia object composition and communication. In: *IEEE Network* 4 (1990), no. 6, pp. 32–40, 45–49

[LG90b] LITTLE, T. D. C.; GHAFOOR, A.: Synchronization and Storage Models for Multimedia Objects. In: *IEEE Journal on Selected Areas in Communications* 8 (1990), no. 3, pp. 413–427

[LG93] LITTLE, T. D. C.; GHAFOOR, A.: Interval-Based Conceptual Models for Time-Dependent Multimedia Data. In: *IEEE Trans. on Knowledge and Data Engineering* 5 (1993), no. 4, pp. 551–563

[LH04] LITTLE, S.; HARDMAN, L.: Cuypers Meets Users: Implementing a User Model Architecture for Multimedia Presentation Generation. In: CHEN, Y.-P. P. (ed.), *10th Int. Multimedia Modeling Conf.; Brisbane, Australia*, IEEE Computer Society Press, Jan. 2004, pp. 364–364

[LJH04] LEOPOLD, K.; JANNACH, D.; HELLWAGNER, H.: Knowledge-based Media Adaptation. In: *Proc. of the SPIE Int. Symposium ITCom 2004 on Internet Multimedia Management Systems IV, Vol. 5601; Philadelphia, PA, USA*, Oct. 2004, pp. 111–120

[LL02] LIANG, T.-P.; LAI, H.-J.: Discovering User Interests from Web Browsing Behavior: An Application to Internet News Services. In: *Proc. of the 35th Annual Hawaii Int. Conf. on System Sciences-Volume 7*, IEEE Computer Society Press, 2002, pp. 203–213

[LL03a] LEMLOUMA, T.; LAYAÏDA, N.: Media Resources Adaptation for Limited Devices. In: *Proc. of the 7th ICCC/IFIP Int. Conf. on Electronic Publishing; Universidade deo Minho, Portugal*, Jun. 2003, pp. 209–218

[LL03b] LEMLOUMA, T.; LAYAÏDA, N.: Adapted Content Delivery for Different Contexts. In: *Symposium on Applications and the Internet; Orlando, FL, USA*, IEEE Computer Society Press, Jan. 2003, pp. 190–197

[LL04] LEMLOUMA, T.; LAYAÏDA, N.: Context-Aware Adaptation for Mobile Devices. In: *IEEE Int. Conf. on Mobile Data Management; Berkeley, CA, USA*, Jan. 2004, pp. 106–111

[LLHC01] LIU, D.; LIN, Y.; HUANG, Y.; CHEN, C.: A Framework for Personalized E-Catalogs: An Integration of XML-based Metadata, User Models and Agents. In: *Proc. of the 34th Annual Hawaii Int. Conf. on System Sciences-Volume 7*, IEEE Computer Society Press, 2001, pp. 70–80

[LN95] LANDIN, N.; NIKLASSON, A.: *Development of Object-Oriented Frameworks*. Diploma thesis, Lund Institute of Technology, Lund University, Lund, Sweden, 1995

[LNK04] LUGMAYR, A.; NIIRANEN, S.; KALLI, S.: *Digital interactive TV and metadata: future broadcast multimedia*. Springer-Verlag, 2004

[LO96] LIN, J.; OZSOYOGLU, Z. M.: Processing OODB queries by O-Algebra. In: *Proc. of the 5th Int. Conf. on Information and knowledge management; Rockville, MD, USA*, ACM Press, 1996, pp. 134–142

[LSB+99] LEE, T.; SHENG, L.; BOZKAYA, T.; BALKIR, N. H.; ÖZSOYOGLU. Z. M.; ÖZSOYOGLU, G.: Querying Multimedia Presentations Based on Content. In: *IEEE Trans. on Knowledge and Data Engineering* 11 (1999), no. 3, pp. 361–385

[LSB+00] LEE, T.; SHENG, L.; BALKIR, N. H.; AL-HAMDANI, A.; ÖZSOYOGLU, G.; ÖZSOYOGLU, Z. M.: Query Processing Techniques for Multimedia Presentations. In: *Multimedia Tools Appl.* 11 (2000), no. 1, pp. 63–99

[MAB00] MULVENNA, M. D.; ANAND, S. S.; BÜCHNER, A. G.: Personalization on the Net using Web mining: introduction. In: *Communications of the ACM* 43 (2000), no. 8, pp. 122–125

[Mac03] MACROMEDIA, INC., USA: Macromedia Flash white paper: Flash Player for Developers and Publishers. Jul. 2003. Retrieved February 8, 2006, from http://www.macromedia.com/software/flash/survey/whitepaper_jul03.pdf

[Mac04] MACROMEDIA, INC., USA: Macromedia Flash (SWF) File Format Specification Version 7. 2004. Retrieved February 8, 2006, from http://www.macromedia.com/software/flash/open/licensing/fileformat/

[Mai03] MAIN, D. N.: JavaSWF2 - A Java Toolkit for Macromedia Flash(TM) Parsing and Generation. Jul. 2003. Retrieved May 12, 2006, from http://www.anotherbigidea.com/javaswf/

[Mar98] MARK, G.: *Patterns in Java: a catalog of reusable design patterns*, vol. 1. John Wiley & Sons, 1998

[Mat96] MATTSSON, M.: *Object-oriented frameworks: a survey of methodological issues*. Licentiate thesis, Department of Computer Science and Business Administration, University College of Karlskrona/Ronneby, Sweden, Feb. 1996

[Mat00] MATTSSON, M.: *Evolution and Composition of Object-Oriented Frameworks*. Karlskrona, Sweden: Kaserntryckeriet AB, Feb. 2000.

PhD thesis, Department of Software Engineering and Computer Science, University of Karlskrona/Ronneby, Sweden

[May93] MAYBURY, M. T. (ed.): *Intelligent multimedia interfaces*. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1993

[MB97] MATTSSON, M.; BOSCH, J.: Framework Composition: Problems, Causes and Solutions. In: *Proc. of the Tools-23: Technology of Object-Oriented Languages and Systems; Santa Barbara, CA, USA*, IEEE Computer Society Press, 1997, pp. 203–214

[McC00] MCCARTHY, J.: Phenomenal data mining. In: *Communications of the ACM* 43 (2000), no. 8, pp. 75–79

[Mei06] MEISTER, J.: *Produktgetriebene Entwicklung von Software-Produktlinien am Beispiel analytischer Anwendungssoftware (Product-driven development of software product lines considering analytical application software as example)*. PhD thesis, Carl von Ossietzky University of Oldenburg, Department of Computing Science, Oldenburg, Germany, 2006

[Mer06a] MERRIAM-WEBSTER, INC., USA: Definition of multimedia. 2006. Retrieved February 27, 2006, from http://www.m-w.com/dictionary/multimedia

[Mer06b] MERRIAM-WEBSTER, INC., USA: Definition of personalize. 2006. Retrieved February 28, 2006, from http://www.m-w.com/dictionary/personalization

[Mey90] MEYER, B.: Lessons from the design of the Eiffel libraries. In: *Communications of the ACM* 33 (1990), no. 9, pp. 68–88

[Mey92] MEYER, B.: Applying Design by Contract. In: *Computer* 25 (1992), no. 10, pp. 40–51

[Mey97] MEYER, B.: *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997

[Mic06a] MICROSOFT CORPORATION, USA: Microsoft DirectShow 9.0. 2006. Retrieved March 6, 2006, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/directshow.asp

[Mic06b] MICROSOFT CORPORATION, USA: Microsoft DirectShow 9.0: AVI File Format. 2006. Retrieved February 7, 2006, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/avifileformat.asp

[MJ92] MCCRAE, R. R.; JOHN, O. P.: An introduction to the five-factor model and its applications. In: *Journal of Personality - Special Issue: The five-factor model: Issues and applications* 60 (1992), pp. 175–215

[MLK+01] METSO, M.; LÖYTYNOJA, M.; KORVA, J.; MÄÄTTÄ, P.; SAUVOLA, J.: Mobile Multimedia Services: Content Adaptation. In: *3rd Int. Conf. on Information, Communications and Signal Processing; Singapore*, 2001

[MLLR99] MERIALDO, B.; LEE, K. T.; LUPARELLO, D.; ROUDAIRE, J.: Automatic construction of personalized TV news programs. In: *Proc. of the 7th ACM Int. Conf. on Multimedia (Part 1); Orlando, FL, USA*, ACM Press, 1999, pp. 323–331

[MMHR04] MATEVSKA-MEYER, J.; HASSELBRING, W.; REUSSNER, R. H.: Software Architecture Description supporting Component Deployment and System Runtime Reconfiguration. In: BOSCH, J.; SZYPERSKI, C.; WECK, W. (ed.), *Proc. of the 9th Int. Workshop on Component-Oriented Programming; Oslo, Norway*, Jun. 2004. Retrieved May 17, 2006, from http://research.microsoft.com/~cszypers/events/WCOP2004/11-Matevska-Hasselbring-Reussner.pdf

[MPE06] MPEG (MOVING PICTURE EXPERTS GROUP): MPEG Home Page. 2006. Retrieved February 7, 2006, from http://www.chiariglione.org/mpeg/

[MPR00] MANBER, U.; PATEL, A.; ROBISON, J.: Experience with personalization of Yahoo! In: *Communications of the ACM* 43 (2000), no. 8, pp. 35–39

[MRT93] MCKEOWN, K.; ROBIN, J.; TANENBLATT, M.: Tailoring lexical choice to the user's vocabulary in multimedia explanation generation. In: *Proc. of the 31st Conf. on Association for Computational Linguistics; Columbus, OH, USA*, 1993, pp. 226–234

[MZ00] MALAKA, R.; ZIPF, A.: DEEP MAP - Challenging IT Research in the Framework of a Tourist Information System. In: FESENMAIER, D.; KLEIN, S.; BUHALIS, D. (ed.), *Proc. of 7th Int. Congress on Tourism and Communications Technologies in Tourism; Barcelona, Spain*, Springer-Verlag, 2000, pp. 15–27

[Neb04] NEBULON PTY. LTD., AUSTRALIA: Feature Driven Development: The portal for all things FDD. 2004. Retrieved March 6, 2006, from http://www.featuredrivendevelopment.com/

[Net94a] NETWORK WORKING GROUP: Uniform Resource Locators (URL). Dec. 1994. Retrieved March 6, 2006, from http://www.w3.org/Addressing/rfc1738.txt

[Net94b] NETWORK WORKING GROUP: Universal Resource Identifiers in WWW. Jun. 1994. Retrieved March 6, 2006, from http://www.w3.org/Addressing/rfc1630.txt

[NGS+01] NICKLAS, D.; GROSSMANN, M.; SCHWARZ, T.; VOLZ, S.; MITSCHANG, B.: A Model-Based, Open Architecture for Mobile, Spatially Aware Applications. In: *Proc. of the 7th Int. Symposium on Advances in Spatial and Temporal Databases; Redondo Beach, CA, USA*, Springer-Verlag, Jul. 2001, pp. 117–135

[Nie98] NIELSEN, J.: Personalization is Over-Rated. Oct. 1998. Retrieved February 28, 2006, from http://www.useit.com/alertbox/981004.html

[Nin96] NING, J. Q.: A Component-Based Software Development Model. In: *20th Computer Software and Applications Conf.; Seoul, Koria*, IEEE Computer Society Press, 1996, pp. 389–395

[OB02] VAN OMMERING, R.; BOSCH, J.: Components in Product-Line Architectures. In: CRNKOVIC and LARSSON [CL02], ch. 11, pp. 207–221

[OCC88] ORTONY, A.; CLORE, G. L.; COLLINS, A.: *The Cognitive Structure of Emotions*. Cambridge University Press, 1988

[OCG+00] VAN OSSENBRUGGEN, J. R.; CORNELISSEN, F. J.; GUERTS, J. P. T. M.; RUTLEDGE, L. W.; HARDMAN, H. L.: *Cuypers: a semi-automatic hypermedia presentation system*. tech. rep. INS-R0025, CWI, The Netherlands, Dec. 2000

[Oes04] OESTERDIEKHOFF, B.: Transcoding von Webinhalten (Transcoding of web content). In: *Informatik Spektrum* 27 (2004), no. 5, pp. 448–452

[OFF06] OFFIS INSTITUTE FOR INFORMATION TECHNOLOGY, OLDENBURG, GERMANY: Competence Center of Lower Saxony for Information Systems for Mobile Usage (Niccimon). 2006. Retrieved March 6, 2006, from http://www.niccimon.de/cms/transform.php?lang=_engl

[OHGR03] VAN OSSENBRUGGEN, J.; HARDMAN, L.; GEURTS, J.; RUTLEDGE, L.: Towards a multimedia formatting vocabulary. In: *Proc. of the 12th Int. Conf. on World Wide Web; Budapest, Hungary*, ACM Press, 2003, pp. 384–393

[OMG05] OMG (OBJECT MANAGEMENT GROUP), INC., USA: Unified Modeling Language Specification: Version 2, Revised Final Adopted Specification (ptc/05-07-04). 2005. Retrieved March 31, 2006, from http://www.uml.org/#UML2.0

[Ope01] OPEN MOBILE ALLIANCE, LA JOLLA, CA, USA: Wireless Markup Language: Version 2.0. Sept. 2001. Retrieved March 27, 2006, from http://www.openmobilealliance.org/tech/affiliates/wap/wap-238-wml-20010911-a.pdf

[Ope06a] OPEN BUSINESS CLUB GMBH, GERMANY: openBC. 2006. Retrieved March 22, 2006, from http://www.openbc.com/

[Ope06b] OPEN MOBILE ALLIANCE, LA JOLLA, CA, USA: OMA User Agent Profile V2.0 Approved Enabler. 2006. Retrieved March 6, 2006, from http://www.openmobilealliance.org/

[Opw96] OPWIS, K.: Ziel (Goal). In: STRUBE et al. [SBF+96], p. 830

[Ora04] ORACLE CORPORATION, USA: interMedia. 2004. Retrieved March 6, 2006, from http://www.oracle.com/technology/products/intermedia/index.html

[Ora06a] ORACLE CORPORATION, USA: interMedia Documentation. 2006. Retrieved March 6, 2006, from http://www.oracle.com/technology/documentation/intermedia.html

[Ora06b] ORATRIX, THE NETHERLANDS: GRiNS for SMIL Homepage. 2006. Retrieved March 6, 2006, from http://www.oratrix.com/GRiNS/

[Ora06c] ORATRIX, THE NETHERLANDS: Oratrix Development. 2006. Retrieved March 6, 2006, from http://www.oratrix.com/

[Par89] PARIS, C. L.: The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise. In: KOBSA and WAHLSTER [KW89], pp. 200–232

[PAS98] PREE, W.; ALTHAMMER, E.; SIKORA, H.: Framelets als handliche Architekturbausteine (Framelets as handy architecture building blocks). In: *Softwaretechnik; Paderborn, Germany*, Sept. 1998

[Pea00] PEARL, J.: *Causality: models, reasoning, and inference*. Cambridge University Press, 2000

[Ped00] PEDNAULT, E. P. D.: Representation is everything. In: *Communications of the ACM* 43 (2000), no. 8, pp. 80–83

[Pey03] PEYTON, L.: Measuring and managing the effectiveness of personalization. In: *Proc. of the 5th Int. Conf. on Electronic commerce; Pittsburgh, PA, USA*, ACM Press, 2003, pp. 220–224

[Pic98] PICARD, R. W.: Toward Agents that Recognize Emotion. In: *Actes Proc. IMAGINA; Monaco*, 1998, pp. 153–165

[Pih03] PIHKALA, K.: *Extensions to the SMIL Multimedia Language*. PhD thesis, Helsinki University of Technology, 2003

[PKK01] POSPISCHIL, G.; KUNCZIER, H.; KUCHAR, A.: LoL@: a UMTS location based service. In: *Int. Symp. on 3rd Generation Infrastructure and Services; Athen, Greece*, 2001

[PLV97] POSNAK, E. J.; LAVENDER, R. G.; VIN, H. M.: An adaptive framework for developing multimedia software components. In: *Communications of the ACM* 40 (1997), no. 10, pp. 43–47

[PR93] PEPPERS, D.; ROGERS, M.: *The One to One Future: Building Relationships One Customer at a Time*. Doubleday, New York, USA, 1993

[Pre94] PREE, W.: Meta Patterns - A Means for Capturing the Essentials of Reusable Object-Oriented Design. In: *Lecture Notes in Computer Science* 821 (1994), pp. 150–162

[Pre95a] PREE, W.: *Design Patterns for Object-Oriented Software Development*. ACM Press Books, Addison-Wesley, 1995

[Pre95b] PREE, W.: State-of-the-art - Design Pattern Approaches: An Overview. In: *Proc. of Technology of Object-Oriented Languages and Systems*, Mar. 1995

[Pre96a] PREE, W.: *Framework Patterns*. SIGS Books and Multimedia, 1996

[Pre96b] PREE, W.: Frameworks - Past, present, future. In: *Object magazine: improving software quality through object development & reuse* 6 (1996), no. 3, pp. 24–36

[Pre97a] PRECHELT, L.: Why We Need an Explicit Forum for Negative Results. In: *Journal of Universal Computer Science* 3 (1997), no. 9, pp. 1074–1083

[Pre97b] PREE, W.: Component-Based Software Development - A New Paradigm in Software Engineering? In: *Software - Concepts and Tools* 18 (1997), no. 4, pp. 169–174

[Pre97c] PREE, W.: Essential Framework Design Patterns. In: *Object magazine: improving software quality through object development & reuse* 7 (1997), no. 1

[Pre97d] PREE, W.: *Komponentenbasierte Softwareentwicklung mit Frameworks (Component-based software development with frameworks)*. dpunkt.verlag, 1997

[Pre97e] PREE, W.: Object-Oriented Design Patterns and Hot Spot Cards. In: *IEEE Int. Conf. on the Engineering of Complex Computer Systems; Como, Italy*, Sept. 1997

[Pre99] PREE, W.: Hot-Spot-Driven Development. In: FAYAD et al. [FSJ99a], pp. 379–393

[PS94] PAPADIAS, D.; SELLIS, T.: Qualitative Representation of Spatial Knowledge in Two-Dimensional Space. In: *The VLDB Journal: The International Journal on Very Large Data Bases* 3 (1994), no. 4, pp. 479–516

[PTOM98] PAULO, F. B.; TURINE, M. A. S.; DE OLIVEIRA, M. C. F.; MASIERO, P. C.: XHMBS: a formal model to support hypermedia specification. In: *Proc. of the 9th ACM Conf. on Hypertext and hypermedia: links, objects, time and space - structure in hypermedia systems; Pittsburgh, PA, USA*, ACM Press, 1998, pp. 161–170

[PTSE95] PAPADIAS, D.; THEODORIDIS, Y.; SELLIS, T.; EGENHOFER, M. J.: Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-Trees. In: *Proc. of the ACM SIGMOD Conf. on Management of Data; San Jose, CA, USA*, Mar. 1995, pp. 92–103

[PUM02] POSPISCHIL, G.; UMLAUFT, M.; MICHLMAYR, E.: Designing LoL@, a Mobile Tourist Guide for UMTS. In: *Proc. of the 4th Int. Symposium on Mobile Human-Computer Interaction; London, UK*, Springer-Verlag, 2002, pp. 140–154

[PVL96] POSNAK, E. J.; VIN, H. M.; LAVENDER, R. G.: Presentation Processing Support for Adaptive Multimedia Applications. In: *Proc. of Multimedia Computing and Networking; San Jose, CA, USA*, Jan. 1996, pp. 234–245

[RB96] RABIN, M. D.; BURNS, M. J.: Multimedia authoring tools. In: *Conf. companion on Human factors in computing systems; Vancouver, British Columbia, Canada*, ACM Press, 1996, pp. 380–381

[RBO+00] RUTLEDGE, L.; BAILEY, B.; VAN OSSENBRUGGEN, J.; HARDMAN, L.; GEURTS, J.: Generating presentation constraints from rhetorical structure. In: *Proc. of the 11th ACM on Hypertext and Hypermedia; San Antonio, TX, USA*, ACM Press, 2000, pp. 19–28

[RBSP02] RIEBISCH, M.; BÖLLERT, K.; STREITFERDT, D.; PHILIPPOW, I.: Extending Feature Diagrams with UML Multiplicities. In: *6th Conf. on Integrated Design and Process Technology; Pasadena, CA, USA*, Jun. 2002, pp. 1–7

[RE99] ROSEL, A.; ERNI, K.: Experiences with the Semantic Graphics Framework. In: FAYAD et al. [FSJ99b], ch. 27, pp. 629–657

[Rea01] REALNETWORKS, INC, USA: RealSystem iQ Production Guide. Sept. 2001. Retrieved February 7, 2006, from http://service.real.com/help/library/guides/productionguidepreview/PDF/productionguide.pdf

[Rea05] REALNETWORKS, INC., USA: Helix Community. 2005. Retrieved March 24, 2006, from https://helixcommunity.org/

[Rea06] REALNETWORKS, INC, USA: RealPlayer. 2006. Retrieved March 6, 2006, from http://www.real.com/

[Rep03] REPPEL, A.: Personalization and the Power to Control. Jun. 2003. Retrieved February 28, 2006, from http://www.reppel.co.uk/relationship-marketing/personalization-and-the-power-to-control.html

[Reu01a] REUSSNER, R. H.: *Parametrisierte Verträge zur Protokolladaption bei Software-Komponenten (Parameterized contracts for adapting*

software component protocols). Logos Verlag, Berlin, Germany, 2001. PhD thesis, Fakultät für Informatik, University Karlsruhe (TH), Germany

[Reu01b] REUSSNER, R. H.: The Use of Parameterised Contracts for Architecting Systems with Software Components. In: WECK, W.; BOSCH, J.; SZYPERSKI, C. (ed.), *Proc. of the 6th Int. Workshop on Component-Oriented Programming; Budapest, Hungary*, Jun. 2001

[RG98] RIEHLE, D.; GROSS, T.: Role model based framework design and integration. In: *Proc. of the 13th ACM SIGPLAN Conf. on Object-oriented programming, systems, languages, and applications; Vancouver, British Columbia, Canada*, ACM Press, 1998, pp. 117–133

[RGB+03] RUMP, N.; GERAMANI, K.; BALDZER, J.; THIEME, S.; SCHERP, A.; KRÖSCHE, J.; MEYER, J.: Potentials of pervasive computing in highly interactive workplaces. In: CHA, J.; JARDIM-GONÇALVES, R.; STEIGER-GARÇÃO, A. (ed.), *10th ISPE Int. Conf. on concurrent Engineering: Research and Applications; Madeira Island, Portugal*, A. A. Balkema Publishers, Jul. 2003, pp. 733–739

[Rhe03] RHEM, A. J. & ASSOCIATES, INC, USA: Knowledge Acquisition Framework. 2003. Retrieved March 6, 2006, from http://www.ajrhem.com/method.html

[RHOB98] RUTLEDGE, L.; HARDMAN, L.; VAN OSSENBRUGGEN, J.; BULTERMAN, D. C. A.: Implementing Adaptability in the Standard Reference Model for Intelligent Multimedia Presentation Systems. In: *Proc. of the 1998 Conf. on MultiMedia Modeling*, IEEE Computer Society Press, 1998, pp. 12–20

[Ric89] RICH, E.: Stereotypes and User Modeling. In: KOBSA and WAHLSTER [KW89], pp. 35–51

[Ric02] RICHTERS, M.: *A Precise Approach to Validating UML Models and OCL Constraints*. Logos, Berlin, Germany, 2002. PhD thesis, University of Bremen, Germany

[Ric03] RICHTER, C.: *Entwurf und Realisierung eines Web-basierten personalisierten multimedia Musik-Newsletters (Design and implementation of a web-based personalized multimedia music newsletter)*. Individual project, Carl von Ossietzky University Oldenburg, Department of Computing Science, Oldenburg, Germany, Jul. 2003

[Rie00a] RIECKEN, D.: Personalized communication networks. In: *Communications of the ACM* 43 (2000), no. 8, pp. 41–42

[Rie00b] RIEHLE, D.: *Framework Design: A Role Modeling Approach*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2000

[RJ97] ROBERTS, D.; JOHNSON, R.: Evolving Frameworks - A Pattern Language for Developing Object-Oriented Frameworks. In: *Pattern Languages of Program Design 3*, Addison-Wesley, 1997

[RJMB93] VAN ROSSUM, G.; JANSEN, J.; MULLENDER, S.; BULTERMAN, D. C. A.: CMIFed: a presentation environment for portable hypermedia documents. In: *Proc. of the 1st ACM Int. Conf. on Multimedia; Anaheim, CA, USA*, ACM Press, 1993, pp. 183–188

[RKB02] RIST, T.; KRÖNER, A.; BRANDMEIER, P.: Layout Adaptation in a Portal for Personalised Cross-Plattform Content Packaging. In: *ABIS-Workshop Personalization for the mobile World; Hanover, Germany*, Oct. 2002, pp. 75–82

[RLHJ99] RAGGETT, D.; LE HORS, A.; JACOBS, I.: HTML 4.01 Specification: W3C Recommendation. Dec. 1999. Retrieved March 6, 2006, from http://www.w3.org/TR/1999/REC-html401-19991224/

[RN03] RUSSELL, S. J.; NORVIG, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003

[RNL95] RAKOW, T. C.; NEUHOLD, E. J.; LÖHR, M.: Multimedia Database Systems - The Notions and the Issues. In: LAUSEN, G. (ed.), *Datenbanksysteme in Büro, Technik und Wissenschaft, Fachtagung der Gesellschaft für Informatik; Dresden, Germany*, Springer-Verlag, Mar. 1995, pp. 1–29

[ROHB98] RUTLEDGE, L.; VAN OSSENBRUGGEN, J.; HARDMAN, L.; BULTERMAN, D. C. A.: Practical application of existing hypermedia standards and tools. In: *Proc. of the 3rd ACM Conf. on Digital libraries; Pittsburgh, PA, USA*, ACM Press, 1998, pp. 191–199

[RS99] ROUT, T. P.; SHERWOOD, C.: Software Engineering Standards and the Development of Multimedia-Based Systems. In: *4th IEEE Int. Symposium and Forum on Software Engineering Standards; Curitiba, Brazil*, May 1999, pp. 192–198

[RS02] REUSSNER, R. H.; SCHMIDT, H. W.: Using Parameterised Contracts to Predict Properties of Component Based Software Architectures. In: CRNKOVIC, I.; LARSSON, S.; STAFFORD, J. (ed.), *Workshop On Component-Based Software Engineering (in association with 9th IEEE Conf. and Workshops on Engineering of Computer-Based Systems); Lund, Sweden*, Apr. 2002, pp. 31–34

[RSA06] Research Studios Austria - Digital Memory Engineering. 2006. Retrieved March 6, 2006, from http://dme.researchstudio.at/

[RSG01] ROSSI, G.; SCHWABE, D.; GUIMARÃES, R.: Designing personalized web applications. In: *Proc. of the 10th World Wide Web Conf.; Hong Kong, Hong Kong*, ACM Press, May 2001, pp. 275–284

[RV97] RESNICK, P.; VARIAN, H. R.: Recommender systems. In: *Communications of the ACM* 40 (1997), no. 3, pp. 56–58

[RWP04] ROSS, K.; WESTERMANN, G. U.; POPITSCH, N.: METIS - A Flexible Database Solution for the Management of Multimedia Assets. In: *Proc. of the 10th Int. Workshop on Multimedia Information Systems; College Park, MD, USA*, Aug. 2004

[Sal89] SALOMAA, A.: *Computation and automata*. Cambridge University Press, 1989

[SAW94] SCHILIT, B.; ADAMS, N.; WANT, R.: Context-Aware Computing Applications. In: *Workshop on Mobil Computing Systems and Applications; Santa Cruz, CA, USA*, IEEE Computer Society Press, 1994, pp. 85–90

[SB92] SINGH, N. N.; BEALE, I. L. (ed.): *Learning Disabilities: Nature, Theory, and Treatment*. Springer-Verlag, 1992

[SB04a] SCHERP, A.; BOLL, S.: Generic Support for Personalized Mobile Multimedia Tourist Applications. In: *Proc. of the 12th annual ACM Int. Conf. on Multimedia, Technical Demonstration; New York, NY, USA*, ACM Press, Oct. 2004, pp. 178–179

[SB04b] SCHERP, A.; BOLL, S.: mobileMM4U - framework support for dynamic personalized multimedia content on mobile systems. In: BRANKI, C.; HAMPE, J. F.; HELFRICH, R.; KURBEL, K.; SCHWABE, G.; TEUTEBERG, F.; UELLNER, S.; UNLAND, R.; WANNER, G. (ed.), *Techniques and Applications for Mobile Commerce (TAMoCo), Multikonferenz Wirtschaftsinformatik; University of Duisburg/Essen, Essen, Germany*, Aka GmbH, Berlin, Germany, Mar. 2004, vol. 3, pp. 204–215

[SB05a] SCHERP, A.; BOLL, S.: A Lightweight Process Model and Development Methodology for Component Frameworks. In: *Proc. of the 10th Int. Workshop on Component-Oriented Programming; Glasgow, Scotland*, Jul. 2005. Retrieved February 8, 2006, from http://research.microsoft.com/~cszypers/events/WCOP2005/

[SB05b] SCHERP, A.; BOLL, S.: Context-driven smart authoring of multimedia content with xSMART. In: *Proc. of the 13th annual ACM Int. Conf. on Multimedia; Hilton, Singapore*, ACM Press, 2005, pp. 802–803

[SB05c] SCHERP, A.; BOLL, S.: MM4U - A framework for creating personalized multimedia content. In: NEPAL, S.; SRINIVASAN, U. (ed.), *Managing Multimedia Semantics*, Idea Group Publishing, ch. 11, 2005, pp. 246–287

[SB05d] SCHERP, A.; BOLL, S.: Paving the Last Mile for Multi-Channel Multimedia Presentation Generation. In: CHEN, Y.-P. P. (ed.), *Proc. of the 11th Int. Conf. on Multimedia Modeling; Melbourne, Australia*, IEEE Computer Society Press, Jan. 2005, pp. 190–197

[SB06] SCHERP, A.; BOLL, S.: Multimedia-Architekturen (Multimedia architectures). In: REUSSNER, R.; HASSELBRING, W. (ed.), *Handbuch der Software-Architektur*, dpunkt.verlag, ch. 21, Feb. 2006, pp. 423–443

[SBF+96] STRUBE, G.; BECKER, B.; FREKSA, C.; HAHN, U.; OPWIS, K.; PALM, G. (ed.): *Wörterbuch der Kognitionswissenschaft (Encyclopedia of cognitive science)*. Stuttgart, Germany: Klett-Cotta, 1996

[SBG99] SCHMIDT, A.; BEIGL, M.; GELLERSEN, H.-W.: There is more to context than location. In: *Computers & Graphics* 23 (1999), no. 6, pp. 893–901

[SBR02] SMYTH, B.; BRADLEY, K.; RAFTER, R.: Personalization techniques for online recruitment services. In: *Communications of the ACM* 45 (2002), no. 5, pp. 39–40

[SC00] SMYTH, B.; COTTER, P.: A personalized television listings service. In: *Communications of the ACM* 43 (2000), no. 8, pp. 107–111

[SCDB04] STASH, N.; CRISTEA, A.; DE BRA, P.: Authoring of Learning Styles in Adaptive Hypermedia. In: *World Wide Web Conf., Education Track; New York, NY, USA*, ACM Press, 2004, pp. 114–123

[Sch97] SCHULMEISTER, R.: Intelligent Tutorial Systems. In: *Hypermedia Learning Systems: Theory - Didactics - Design*, Hamburg, Germany: Oldenbourg, Munich, Germany, ch. 6, 2nd edition, 1997. Retrieved March 6, 2006, from http://www.izhd.uni-hamburg.de/paginae/Book/default.html

[Sch99] SCHMID, H. A.: Framework Design by Systematic Generalization. In: FAYAD et al. [FSJ99a], ch. 15, pp. 353–378

[Sch00] SCHMIDT, A.: Implicit Human Computer Interaction Through Context. In: *Personal Technologies* 4 (2000), no. 2&3, pp. 191–199

[Sch02] SCHERP, A.: Software Development Process Model and Methodology for Virtual Laboratories. In: HAMZA, M. H. (ed.), *Proc. of the IASTED Int. Conf. on Applied Informatics, Int. Symposium on Software Engineering, Databases, and Applications; Innsbruck, Austria*, ACTA Press, Anaheim, Calgary, Zurich, Feb. 2002, pp. 47–52

[SDB03] STASH, N.; DE BRA, P.: Building Adaptive Presentations with AHA! 2.0. In: *Proc. of the PEG Conf.; Sint Petersburg, Russia*, Jun. 2003

[Sea06] SEARCHNETWORKING.COM, USA: Glossary - Definition - location-based services. 2006. Retrieved March 6, 2006, form http://searchnetworking.techtarget.com/gDefinition/0,294236,sid7_gci532097,00.html

[SG96] SHAW, M.; GARLAN, D.: *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996

[SGM02] SZYPERSKI, C.; GRUNTZ, D.; MURER, S.: *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 2nd edition, 2002

[SHGN04] SCHWARZ, T.; HÖNLE, N.; GROSSMANN, M.; NICKLAS, D.: A Library for Managing Spatial Context Using Arbitrary Coordinate Systems. In: *Workshop on Context Modeling and Reasoning, Proc. of the 2nd IEEE Conf. on Pervasice Computing and Communications; Orlando, FL, USA*, IEEE Computer Society Press, Mar. 2004, pp. 48–52

[Sih01] SIHLING, M.: *Methodische Entwicklung und rollenbasierte Integration von Komponentenframeworks (Methodical development and role-based integration of component frameworks)*. PhD thesis, Technical University Munich, Munich, Germany, 2001

[SKR01] SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J.: E-Commerce Recommendation Applications. In: *Data Min. Knowl. Discov.* 5 (2001), no. 1-2, pp. 115–153

[SMI05] Synchronized Multimedia Integration Language (SMIL 2.1): W3C Recommendation. Dec. 2005. Retrieved February 10, 2006, from http://www.w3.org/TR/2005/REC-SMIL2-20051213/

[SMMW05] SUCHOMSKI, M.; MILITZER, M.; MEYER-WEGENER, K.: RETAVIC: using meta-data for real-time video encoding in multimedia servers. In: *Proc. of the Int. workshop on Network and operating systems support for digital audio and video; Stevenson, WA, USA*, ACM Press, 2005, pp. 81–86

[SN95] STEINMETZ, R.; NAHRSTEDT, K.: *Multimedia: Computing, Communications and Applications*. Prentice Hall, 1995

[Sne01] SNEKKENES, E.: Concepts for personal location privacy policies. In: *Proc. of the 3rd ACM Conf. on Electronic Commerce; Tampa, FL, USA*, ACM Press, 2001, pp. 48–57

[Som04] SOMMERVILLE, I.: *Software Engineering*. Pearson and Addison-Wesley, 7th edition, 2004

[Spi00] SPILIOPOULOU, M.: Web usage mining for Web site evaluation. In: *Communications of the ACM* 43 (2000), no. 8, pp. 127–134

[SRM97] Special Issue of Comput. Stand. Interfaces on Standard Reference Model for Intelligent Multimedia Presentation Systems. Dec. 1997. Vol. 18, no. 6-7

[Sub98] SUBRAHMANIAN, V. S.: Multimedia Query and Presentation Algebras. In: *Proc. of the 9th Int. Workshop on Database and Expert Systems Applications; Washington, DC, USA*, IEEE Computer Society Press, 1998, p. 79

[Sum06] SUMTOTAL SYSTEMS, INC., USA: ToolBook. 2006. Retrieved May 22, 2006, from http://www.toolbook.com/

[Sun99] SUN MICROSYSTEMS, INC., USA: Code Conventions for the Java Programming Language [Reference manual]. Apr. 1999. Retrieved March 6, 2006, from http://java.sun.com/docs/codeconv/

[Sun04] SUN MICROSYSTEMS, INC., USA: How to Write Doc Comments for the Javadoc Tool [Reference manual]. 2004. Retrieved March 6, 2006, from http://java.sun.com/j2se/javadoc/writingdoccomments/

[Sun06] SUN MICROSYSTEMS, INC., USA: Java Technology. 2006. Retrieved March 6, 2006, from http://java.sun.com/

[SYS98] SCHMITZ, P.; YU, J.; SANTANGELI, P.: Timed Interactive Multimedia Extensions for HTML (HTML+TIME) - Extending SMIL into the Web Browser. Sept. 1998. Retrieved March 6, 2006, from http://www.w3.org/TR/NOTE-HTMLplusTIME

[Szy00] SZYPERSKI, C.: Component Software and the Way Ahead. In: LEAVENS, G. T.; SITARAMAN, M. (ed.), *Foundations of component-based systems*, Cambridge University Press, 2000, pp. 1–20

[Tan98] TANNENBAUM, R. S.: *Theoretical Foundations of Multimedia*. W. H. Freeman & Co., New York, NY, USA, 1998

[Tec02] TECHNICAL STANDARDIZATION COMMITTEE ON AV & IT STORAGE SYSTEMS AND EQUIPMENT: Exchangeable image file format for digital still cameras: Exif Version 2.2. Apr. 2002. Retrieved February 7, 2006, from http://www.exif.org/Exif2-2.PDF

[Tec06] TECHNICAL UNIVERSITY OF VIENNA, AUSTRIA: VizIR project webserver. 2006. Retrieved March 6, 2006, from http://vizir.ims.tuwien.ac.at/

[The06a] THE APACHE SOFTWARE FOUNDATION, USA: Apache Tomcat. 2006. Retrieved April 27, 2006, from http://tomcat.apache.org/

[The06b] THE APACHE SOFTWARE FOUNDATION, USA: Log4j project - Introduction. 2006. Retrieved April 18, 2006, from http://logging.apache.org/log4j/docs/

[The06c] THE J. PAUL GETTY TRUST, USA: Getty Thesaurus of Geographic Names. 2006. Retrieved January 26, 2006, from http://www.getty.edu/research/conducting_research/vocabularies/tgn/

[Top02] TOPLAND, K. O.: *Mobile learning: Technological challenges on multi-channel e-learning services*. Masters thesis, Agder University College, Grimstad, Norway, May 2002

[TTC95] TAYLOR, R. N.; TRACZ, W.; COGLIANESE, L.: Software development using domain-specific software architectures: CDRl A011 - a curriculum module in the SEI style. In: *SIGSOFT Softw. Eng. Notes* 20 (1995), no. 5, pp. 27–38

[Uni01] UNIDEX, INC., USA: Universal Turing Machine in XSLT. Mar. 2001. Retrieved March 22, 2006, from http://unidex.com/turing/utm.htm

[Uni06] UNIVERSITY OF BREMEN, GERMANY: A UML-based Specification Environment. 2006. Retrievied May 12, 2006, from http://www.db.informatik.uni-bremen.de/projects/USE/

[UPNM03] UMLAUFT, M.; POSPISCHIL, G.; NIKLFELD, G.; MICHLMAYR, E.: LoL@, a Mobile Tourist Guide for UMTS. In: *Information Technology & Tourism* 5 (2003), no. 3, pp. 151–164

[VAB+00] VERMEULEN, A.; AMBLER, S. W.; BUMGARDNER, G.; METZ, E.; MISFELDT, T.; SHUR, J.; THOMPSON, P.: *The Elements of Java Style*. Cambridge University Press, 2000

[Vil01] VILLARD, L.: Authoring Transformations by direct manipulation for Adaptable Multimedia Presentations. In: *Proc. of the ACM Symposium on Document Engineering; Atlanta, GA, USA*, ACM Press, Nov. 2001, pp. 125–134

[VOGME02] VILLANOVA-OLIVER, M.; GENSEL, J.; MARTIN, H.; ERB, C.: Design and Generation of Adaptable Web Information Systems with KIWIS. In: *Int. Symposium on Information Technology; Las Vegas, NV, USA*, IEEE Computer Society Press, Apr. 2002, pp. 306–311

[VZ02] VOGEL, O.; ZDUN, U.: Content Conversion and Generation on the Web: A Pattern Language. In: *Proc. of European Conf. on Pattern Languages of Programs; Irsee, Germany*, Jul. 2002, pp. 195–232

[W3C99] W3C (WORLD WIDE WEB CONSORTIUM): XSL Transformations (XSLT) Version 1.0: W3C Recommendation. Nov. 1999. Retrieved February 8, 2006, from http://www.w3.org/TR/xslt

[W3C03] W3C (WORLD WIDE WEB CONSORTIUM): Portable Network Graphics (PNG) Specification (Second Edition): Information technology - Computer graphics and image processing - Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003 (E) - W3C Recommendation. Nov. 2003. Retrieved February 7, 2006, from http://www.w3.org/TR/PNG/

[W3C04] W3C (WORLD WIDE WEB CONSORTIUM): Extensible Markup Language (XML) 1.0 (Third Edition) - W3C Recommendation. Feb. 2004. Retrieved February 8, 2006, from http://www.w3.org/TR/2004/REC-xml-20040204/

[W3C05] W3C (WORLD WIDE WEB CONSORTIUM): XML Schema. 2005. Retrived March 6, 2006, from http://www.w3.org/XML/Schema

[W3C06a] W3C (WORLD WIDE WEB CONSORTIUM): Cascading Style Sheets. 2006. Retrieved March 24, 2006, from http://www.w3.org/Style/CSS/

[W3C06b] W3C (WORLD WIDE WEB CONSORTIUM): JPEG JFIF. Feb. 2006. Retrieved February 7, 2006, from http://www.w3.org/Graphics/JPEG/

[Wah02] WAHLSTER, W.: SmartKom: Fusion and Fission of Speech, Gestures, and Facial Expressions. In: *Proc. of the 1st Int. Workshop on Man-Machine Symbiotic Systems; Kyoto, Japan*, Nov. 2002, pp. 213–225

[WB01] WEBER, G.; BRUSILOVSKY, P.: ELM-ART: An Adaptive Versatile System for Web-based Instruction. In: *Int. Journal of Artificial Intelligence in Education* 12 (2001), pp. 351–384

[WBJ90] WIRFS-BROCK, R. J.; JOHNSON, R. E.: Surveying current research in object-oriented design. In: *Communications of the ACM* 33 (1990), no. 9, pp. 104–124

[WDB02] WU, H.; DE BRA, P.: Link-Independent Navigation Support in Web-Based Adaptive Hypermedia. In: *11th Int. World Wide Web Conf.; Honolulu, HI, USA*, May 2002. Retrieved March 21, 2006, from http://www2002.org/CDROM/alternate/684/

[Web06] WEB3D CONSORTIUM, SAN FRANCISCO, CA, USA: X3D Specifications. 2006. Retrieved May 15, 2006, from http://www.web3d.org/x3d/specifications/

[Wec97] WECK, W.: Independently Extensible Component Frameworks. In: MÜHLHÄUSER, M. (ed.), *Special Issues in Object-Oriented Programming*, Heidelberg, Germany: dpunkt.verlag, 1997, pp. 177–183

[Wik06] WIKIPEDIA: WAV. 2006. Retrieved February 7, 2006, from http://en.wikipedia.org/wiki/WAV

[Wil99] WILSON, I.: Artificial Emotion: Simulating Mood and Personality. May 1999. Retrieved February 20, 2006, from http://www.gamasutra.com/features/19990507/artificial_emotion_01.htm

[Wit99] WITTIG, H.: *Intelligent Media Agents: Key technology for Interactive Television, Multimedia and Internet Applications*. GWV-Vieweg, Brunswick, Germany, 1999

[WK89] WAHLSTER, W.; KOBSA, A.: User models in Dialog Systems. In: KOBSA and WAHLSTER [KW89], pp. 4–34

[WKDB01] WU, H.; DE KORT, E.; DE BRA, P.: Design issues for general-purpose adaptive hypermedia systems. In: *Proc. of the 12th ACM Conf. on Hypertext and Hypermedia; Århus, Denmark*, ACM Press, 2001, pp. 141–150

[WMS05] *3rd Special Workshop on Multimedia Semantics; Pisa, Italy*, Jun. 2005. Retrieved March 3, 2006, from http://www.kettering.edu/~pstanche/wms05homepage/wms05.html

[WR94] WAHL, T.; ROTHERMEL, K.: Representing Time in Multimedia Systems. In: *Proc. IEEE Int. Conf. on Multimedia Computing and Systems; Boston, MA, USA*, May 1994, pp. 538–543

[WRB01] WAHLSTER, W.; REITHINGER, N.; BLOCHER, A.: SmartKom: Multimodal Communication with a Life-Like Character. In: *Proc. of*

7th European Conf. on Speech Communication and Technology; Aalborg, Denmark, 2001, pp. 1547–1550

[WRBS05] WECK, W.; REUSSNER, R.; BOSCH, J.; SZYPERSKI, C.: WS17. The Tenth Int. Workshop on Component-Oriented Programming. In: *10th Int. Workshop on Component-Oriented Programming; Glasgow, Scotland*, Jul. 2005. Retrieved March 6, 2006, from http://research.microsoft.com/∼cszypers/events/WCOP2005/

[WS96] WECK, W.; SZYPERSKI, C.: Do we need inheritance? In: *Workshop on Composability Issues in Object-Orientation; Linz, Austria*, Jun. 1996

[WSDSS96] WILLRICH, R.; SÉNAC, P.; DIAZ, M.; DE SAQUI-SANNES, P.: A Formal Framework for the Specification, Analysis and Generation of Standardized Hypermedia Documents. In: *Int. Conf. on Multimedia Computing and Systems; Hiroshima, Japan*, Jun. 1996, pp. 399–406

[WW00] WELLS, N.; WOLFERS, J.: Finance with a personalized touch. In: *Communications of the ACM* 43 (2000), no. 8, pp. 30–34

[WXC+05] WANG, J.; XU, C.; CHNG, E.; DUAN, L.; WAN, K.; TIAN, Q.: Automatic generation of personalized music sports video. In: *Proc. of the 13th annual ACM Int. Conf. on Multimedia; Hilton, Singapore*, ACM Press, 2005, pp. 735–744

[X-S06] X-SMILES: X-Smiles.org. 2006. Retrieved March 6, 2006, http://www.x-smiles.org/

[XFo06] XForms 1.0 (Second Edition): W3C Recommendation. Mar. 2006. Retrieved May 2, 2006, from http://www.w3.org/TR/2006/REC-xforms-20060314/

[XHT05] XHTML 2.0: W3C Working Draft 27. May 2005. Retrieved March 6, 2006, from http://www.w3.org/TR/xhtml2/

[XZM02] XIE, X.; ZENG, H.-J.; MA, W.-Y.: Enabling personalization services on the edge. In: *Proc. of the 10th ACM Int. Conf. on Multimedia; Juan-les-Pins, France*, ACM Press, 2002, pp. 263–266

[Yah06] YAHOO!, INC., USA: MyYahoo! 2006. Retrieved March 6, 2006, from http://my.yahoo.com/

[Zip02] ZIPF, A.: User-Adaptive Maps for Location-Based Services (LBS) for Tourism. In: WOEBER, K.; FREW, A.; HITZ, M. (ed.), *Proc. of the 9th Int. Conf. for Information and Communication Technologies in Tourism; Innsbruck, Austria*, 2002

[Zül05] ZÜLLIGHOVEN, H. (ed.): *Object-Oriented Construction Handbook: Developing Application-Oriented Software with the Tools and Materials Approach*. dpunkt.verlag, 2005

Index

In this index a page number in **semi-bold** font refers to an introduction and definition of a term or concept. Pages marked in normal font contain a term or concept and provide additional information to it.

About the Author

Ansgar Scherp—born in Vechta, Germany on December 15, 1974—studied computer science at the Carl von Ossietzky University of Oldenburg, Germany from 1995 to 2001. He finished his studies with the diploma thesis "Process Model and Development Methodology for Virtual Laboratories" and started working in 2001 as scientific employee at the University of Oldenburg in a project developing methods and tools for virtual laboratories. With a laboratory for genetics engineering, he won (together with M. Schlattmann, A. Hasler, W. Heuten, and R. Kuczewski) the audience award of the MEDIDA-PRIX contest for media-didactics in higher education in 2002. From 2003 to 2006, he has been with the OFFIS Institute for Information Technology, working as scientific employee in the R&D division Multimedia and Internet Information Systems on a project called MM4U (short for "MultiMedia For You"). The aim of this project is to develop a software framework for the dynamic generation of personalized multimedia content. Within the MM4U project, Ansgar Scherp wrote this PhD thesis about the conceptual design and the software engineering issues of the MM4U framework. He received his doctoral degree with distinction in August 2006.