INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Usage-Based Storyboarding for Web Information Systems

Klaus-Dieter Schewe and Bernhard Thalheim

Bericht Nr. 0613
November 2006

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Usage-Based Storyboarding for Web Information Systems

Klaus-Dieter Schewe and Bernhard Thalheim

Bericht Nr. 0613
November 2006

e-mail: k.d.schewe@massey.ac.nz,
thalheim@is.informatik.uni-kiel.de

Dieser Bericht ist als persönliche Mitteilung aufzufassen.[a]

[a]Massey University, Information Science Research Centre
Private Bag 11222, Palmerston North, New Zealand

Abstract

On a high level of abstraction a Web Information System (WIS) can be described by a storyboard, which in an abstract way specifies who will be using the system, in which way and for which goals. While syntax and semantics of storyboarding has been well explored, its pragmatics has not. This paper contributes the first step towards closing this gap by analysing the usage of WISs. Starting from a classification of intentions we first present *life cases*, which capture observations of user behaviour in reality. We discuss the facets of life cases and present a semi-formal way for their documentation. Life cases can be used in a pragmatic way to specify a story space, which is an important component of a storyboard. In a second step we complement life cases by *user models* that are specified by various facets of actor profiles that are needed for them. We analyse actor profiles and present a semi-formal way for their documentation. We outline how these profiles can be used to specify actors, which are an important component of a storyboard. Finally, we analyse *contexts* and the way they impact on life cases, user models and the storyboard.

Keywords: web information system, storyboarding, intentions, pragmatics, life case, context modelling, user modelling

Table of Contents

Chapter 1

Introduction

A Web Information System (WIS) is an information system that can be accessed through the world-wide-web. On a high level of abstraction a WIS can be described by a storyboard [41], which in an abstract way specifies who will be using the system, in which way and for which goals. In a nutshell, a *storyboard* consists of three parts:

- a *story space*, which itself consists of a hierarchy of labelled directed graphs called *scenarios*, one of which is the main scenario, whereas the others define the details of *scenes*, i.e. nodes in a higher scenario, and a *plot* that is specified by an assignment-free process, in which the basic actions correspond to the labels of edges in the scenarios,
- a set of *actors*, i.e. abstractions of user groups that are defined by *roles*, which determine obligations and rights, and *user profiles*, which determine user preferences,
- and a set of *tasks* that are associated with *goals* the users may have.

In addition, there are many constraints comprising static, dynamic and deontic constraints for pre- and postconditions, triggering and enabling events, rights and obligations of roles, preference rules for user types, and other dependencies on the plot. Details of storyboarding have been described in [41]. An overview of our method for the design of WISs was presented in [40].

While syntax and semantics of storyboarding has been well explored, its pragmatics apart from the use of metaphors [47] has not. Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. This relationship may be pictured by the so-called semiotics triangle. Main branches of semiotics are *syntactics*, which is concerned with the syntax, i.e. the construction of the language, *semantics*, which is concerned with the interpretation of the words of the language, and *pragmatics*, which is concerned with the current use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment. Most languages defined in Computer Science have a well-defined syntax; some of them possess a well-defined semantics; few of them use pragmatics through which the meaning might be different for different users.

Syntactics is often based on a constructive or generative approach: Given an alphabet and an set of constructors, the language is defined as the set of expressions that can be generated by the constructors. Constructions may be defined on the basis of grammatical rules.

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g. used for defining the semantics of predicate logics, by procedural or referential semantics, e.g. operational semantics used for defining the semantics of programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

Pragmatics has to be distinguished from pragmatism. Pragmatism means a practical approach to problems or affairs. According to Webster [51] pragmatism is the “balance between principles

and practical usage". Here we are concerned with pragmatics, which is based on the behaviour and demands of users, therefore depends on the understanding of users.

The six characteristics of WISs that were discussed in [41] can be mapped to conceptual structures that are used for storyboard specification:

1. We start with the characteristics used for the strategic layer. Main specification elements used are intention and mission. They are mapped to metaphors, general goals, rhetorical figures, and patterns and grids of web pages discussed later.
2. The scenarios reflect the utilisation by actors, for which we envision a number of stories that correspond to real use. These scenarios may be captured through observation of reality. Story spaces and plots are recorded in various level of detail through the methods discussed in [41]. The stories are reflected in the storyboard.
3. Content specification is the basis for the media types, i.e. data types and their functions, which will be introduced in part III. It combines data specification with user requirements and is reflected in the content portfolio.
4. Functionality is provided by the media types as required by the storyboard. Typical standard functions are navigation, retrieval (search), support functions, and feedback facilities.
5. Context is based on tasks, history, and environment. We use the specification of context for restructuring and functionality enhancement, which will form the basis of XSL transformations and the onion approach that is discussed in the last part of the book.
6. Presentation depends on the intention, the provider, the technical environment available and the users the WIS is targeting at. Presentation results in the layout and the playout of the WIS. *Layout* requires the development of multimedia presentations for each page. *Playout* additionally requires the development of functionality that supports visits of users depending on the story they are currently following to achieve their goals. Layout and playout integrate the chosen metaphors; they depend on chosen page patterns and grids as well as on quality requirements.

Conceptual structures and their association are depicted in Figure 1.1. We may separate the syntactics and pragmatics layers. Arrows are used for representing part-of- or uses- or relates-associations. For instance, the story is based on the user and the functions. Information metaphors relate content to information.

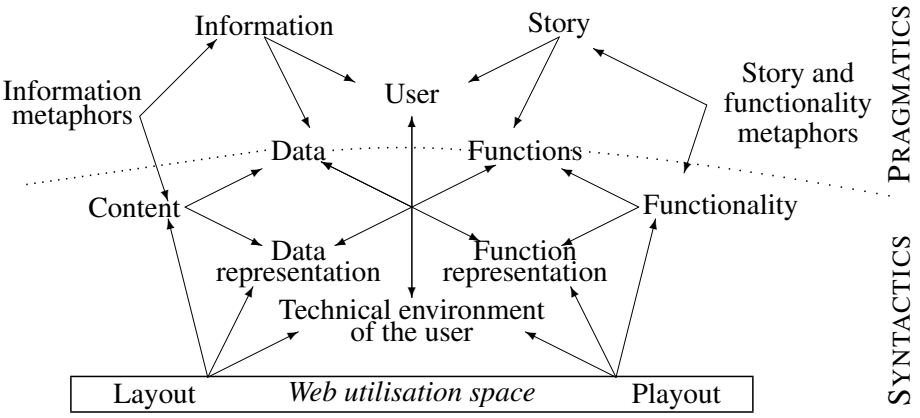


Fig. 1.1. The Web Utilization Space Based On the Characteristics of WIS

We use the notions of information and content in a specific manner. *Information* as processed by humans, is carried by *data* that is perceived or noticed, selected and organized by its receiver,

because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom, simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge. Content is complex and ready-to-use data. Content management systems are information systems that support extraction, storage and delivery of complex data. Content may be enhanced by *concepts* that specify the semantic meaning of content objects and by *topics* that specify the pragmatic understanding of users.

Therefore, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. In order to ease perception we use *metaphors*. Metaphors may be separated into those that support perception of information and into those that support usage or functionality.

The *information transfer* from a user A to a user B depends on the users A and B, their abilities to send and to receive the data, to observe the data, and to interpret the data. Let us formalise this process. Let s_X denote the function used by a user X for data extraction, transformation, and sending of data. Let r_X denote the corresponding function for data receival and transformation, and let o_X denote the filtering or observation function. The data currently considered by X is denoted by D_X. Finally, data filtered or observed must be interpreted by the user X and integrated into the knowledge K_X a user X has. Let us denote by i_X the binary function from data and knowledge to knowledge. By default, we extend the function i_X by the time t_{i_X} of the execution of the function.

Thus, the data transfer and information reception (or briefly information transfer) is formally expressed it by

$$I_B \;=\; i_B(o_B(r_B(s_A(D_A))), K_B, t_{i_X}) \;.$$

In addition, time of sending, receiving, observing, and interpreting can be taken into consideration. In this case we extend the above functions by a time argument. The function s_X is executed at moment t_{s_X}, r_X at t_{r_X}, and o_X at t_{o_X}. We assume $t_{s_A} \leq t_{r_B} \leq t_{o_B} \leq t_{i_B}$ for the time of sending data from A to B. The time of a computation f or data consideration D is denoted by t_f or t_D, respectively. In this extended case the information transfer is formally expressed it by

$$I_B \;=\; i_B(o_B(r_B(s_A(D_A, t_{s_A}), t_{r_B}), t_{o_B}), K_B, t_{i_B}) \;.$$

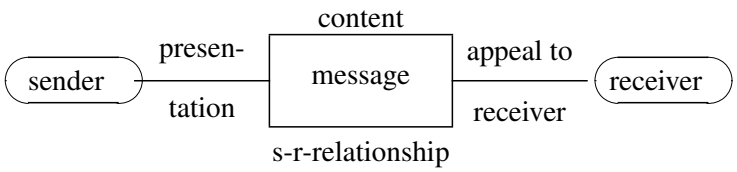


Fig. 1.2. Dimensions of understanding messages

The notion of information extends the dimensions of understanding of message displayed in Figure 1.2 to a web communication act that considers senders, receivers, their knowledge and experience. Figure 1.3 displays the multi-layering of communication, the influence of explicit knowledge and experience on the interpretation.

The communication act is specified by

- the communication message with the content or content chunk, the characterisation of the relationship between sender and receiver, the data that are transferred and may lead to information or misinformation, and the presentation,

– the sender, the explicit knowledge the sender may use, and the experience the sender has, and
– the receiver, the explicit knowledge the receiver may use, and the experience the receiver has.

The communication act is the basis for *sagas* that are the basic units of stories.

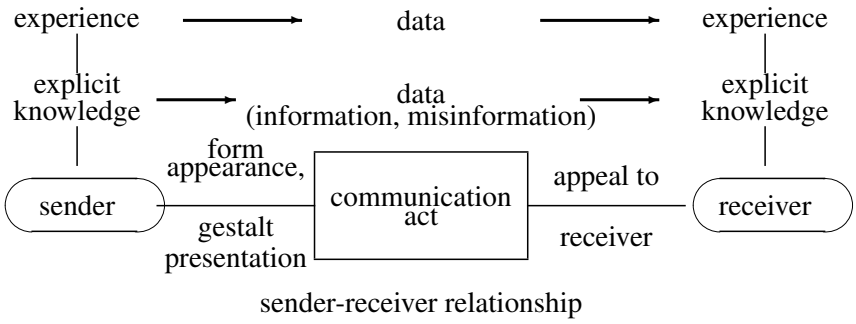


Fig. 1.3. Dimensions of the communication act

Users are reflected by actors that are abstractions of groups of users. Pragmatics and syntactics share data and functions. The functionality is provided through functions and their representations. The web utilisation space depends on the technical environment of the user. It is specified through the layout and the playout. Layout places content on the basis of a data representation and in dependence of the technical environment. Playout is based on functionality and function representations, and depends on the technical environment.

We may base the pragmatics of storyboarding on two ingredients that are easy to develop:

The *utilisation portfolio* of the WIS: The utilisation portfolio consists of

– tasks users might wish to accomplish within the system,
– goals of user groups, i.e. actors, and
– actors that might use the web information system.

The *profiles* of the users: Profiles of users describe their abilities, skills, knowledge, and other dimensions. We map user profiles to preference rules for utilisation of systems.

We first consider these basic questions. They are related to the strategic and the business layers, so they also have an impact on the storyboard. Refinement and utility take us to the conceptual layer.

In this article we approach the analysis of WIS usage as the first important part of storyboarding pragmatics. After a brief review of related work in the literature in Chapter 2 we first look at intentions that are associated with the WIS, and classify them using facets for purpose, intent, time and representation, the first of which was already discussed in the strategic model of WISs [28]. These facets of intentions form the basis for the following three chapters, in which we introduce life cases, user models and contexts. Chapter 4 is devoted to the discussion of *life cases*, which capture observations of user behaviour in reality. We discuss the facets of life cases that impact on the design of storyboards and present a semi-formal way for their documentation. Life cases can be used in a pragmatic way to specify the story space. The work on life cases extends a previous conference publication [39]. In Chapter 5 we complement life cases by *user models* that are specified by user and actor profiles, and actor portfolios. We analyse actor profiles and present a semi-formal way for their documentation. The actor portfolios are used to get a better understanding of the tasks associated with the WIS. The work on user models extends a previous conference publication [42]. Finally, in Chapter 6 we analyse *contexts* and the way they impact on life cases, user models and the storyboard. In Chapter 8 we conclude with a brief summary and a discussion of open issues.

Chapter 2

Related Work

Storyboarding and also the preceding strategic modelling of WIS [28] are unique to our approach to WIS modelling. Other approaches to WIS engineering such as the object oriented OOHDM [16, 37, 45], WebML [9], HERA [21] and variants of UML [10, 26] concentrate on providing models of content, navigation and interaction by means of extended views, which in our own work is captured by so-called *media types* [41]. WSDM [13] emphasises the additional need for a mission statement and a high-level description of processes and users in the WIS. Quite often high-level modelling of WISs is subject to UML-based methods, in particular variants of use cases. The rationale underlying our work is that this is far too little to capture strategic and usage issues of WISs.

The integration of goals and soft goals into the information systems development process has been proposed by [29, 14]. The detailed distinction of intentions as targets, objects, objectives and aims is based on linguistics [51]. The integration of the temporal dimension is necessary because of the information systems context. The extension by the representational dimensions has been made in order to specify aspects of WISs.

Life cases extend and formalise scenarios used in software engineering and user modelling [12]. Early work like already discovered that human computer interfaces can be developed in techniques of movie writing or play development [50]. Life cases have already been envisioned in [7]. Scenarios as defined in software engineering are examples of interaction sessions. This kind of scenario starts with an outline of the interaction, and during requirements elicitation, details are added to create a complete description of interaction. They are specified by a general description of the start and termination states of the system, a description of the normal flow of events, and a description of concurrent events [46]. Later scenario research has been integrated into agile programming. Customer involvement is one of the requirement of agile methods. Customers should be integrated as early as only possible into the development process [3]. Both approaches are to much oriented towards systems and do not consider life cases as we do. Scenario development has been applied to development of websites in [36].

In [17] life cases were integrated into the entire software engineering process. Recently, software engineering research developed an engineering approach to simple life case. Business use cases [35] generalise the requirements shell of [34] that is based on verbal description of event/use cases, requirement types, the rationale, originator, fit criterion, customer (dis)satisfaction, supporting materials, and the history. Our life case specification is not as complex as these business use cases. They combine intention, users, actors, constraints/assumptions/scope or context, tasks, migration, risks, costs, documentation, testing, and requirements for functions, data, look and feel, usability, humanity, operating, maintenance, support, security, culture, politics, legal and performance. We concentrate on the requirements the user has as such.

Our approach to development of systems based on task discovery is based on [32]. Another

task description is used in [11]. This task pre-model is one of the basic ingredients for use cases. Unfortunately, both task models did not get formal underpinnings. Task descriptions are also used in participatory design [8, 23, 31]. Scenarios are also popular in HCI [4]. In this case they represent an envisioned task from a users perspective.They describe users interaction with a system in a concrete narrative form. They allow a walk-trough a system to see how much people have to remember and how many things they will have to search.

Explicit modelling of users has a long tradition in artificial intelligence, and has emerged as a field in its own right [19, 24, 25, 27]. However, the focus is usually slightly different from ours, as we do not intend to adapt system behaviour to observed user behaviour, but use classification of users as a modelling means for capturing WIS pragmatics. This is similar to the focus in [29], though we focus on WISs.

A successful implementation of user profile adaptation for database design and development systems RADD has led to a sophisticated user model [2] that is the basis for our user model. Recently the formal specification of user profiles has gained more attention in the research community. The UserML proposal [18] and user modelling based on ontologies [33]. The general user model ontology GUMO [19] is a trial to systematise the variety of user models.

Contextual modelling has found its own research community [6], but very little work has been done to integrate contextual modelling into WIS design. The work in [30] uses contexts as an approach to modularise conceptual models. The most advanced attempt to modelling context is the work on contextual information bases (CIB) [1, 48, 49]. Roughly speaking, a CIB-context associates objects with a name and an optional reference to another CIB-context. Thus, both the name and the reference depend on the usage history. As shown in [22] CIB-contexts in a slightly generalised form can be combined with media types to provide a formal model of usage context. For this, a location, i.e. a complex value, is associated with media types instead of only associating merely a name with a location L.

Software system engineering separates development into different phases such as requirements analysis and elicitation, software specification, programming and implementation, deployment, etc. The early phases are however not taken into consideration. Recently, business use cases have been introduced [35]. Very few papers and book consider these early phases. [5] and [20] consider application domain description beside [35]. [5] considers software engineering on the basis of the triptych consisting of the application domain description, the requirements prescriptions, and finally the systems specifications.

Chapter 3

Facets of Intention

The description of intention is based on a clear understanding of aims and targets of the WIS including a specification of long-range and short-term targets, expected visitors, characteristics of this audience, tasks performed by the users, necessary and unnecessary content, and finally an understanding of restrictions of usage.

Utilisation scenarios are developed on the basis of intentions. An intention specifies what one purposes to accomplish or do, i.e. one has in mind to do or bring about. It has four facets that are based on the general characteristics and questions discussed in [28]:

Purpose facet: The *purpose* of stakeholders specifies an anticipated outcome that is intended or that guides the planned actions. It may be based on two additional pieces of information:

- The *aims* specify what is intended to be attained and what is believed to be attainable.
- The *objectives* are more general than the aims. They specify something toward which an effort is directed, e.g. goals or ends of an action.

The purpose is already specified at the strategic layer. It depends on the *audience* intended for the WIS, and is influenced by the *mission* that is determined by the WIS provider.

Intent facet: The *intent* suggests a clearer formulation or greater deliberateness. It distinguishes between the following two aspects:

- The *targets* of stakeholders specify the steps of a plan and its termination or satisfaction conditions.
- The *object* of stakeholders is related to wishes or needs of users, and specifies an effort or activity of a user in order to satisfy the need.

The intent facet is related to *tasks* the user wants to accomplish. The intent may be ordered into *major intents* specifying the main uses of the system and *minor intents* that may be only partially of interest. Intents may be generalized to *themes* that represent classes of intents.

Time facet: The *time* facet is used for specification of the general time restrictions such as

- the *design*, which implies a more carefully calculated plan, and
- the *end*, which stresses the intended effect of an action, often in distinction or contrast to the action or means as such.

Time facets may be very general. In this case, we use *occasions* to represent an entire class of time frames.

Representation facet: Intentions are described or annotated through utterances, words or pictures. The word representation is related to word fields used in the strategic model [28]. The icon representation is based on metaphors. Word fields may be specialised to concept fields discussed

later in this chapter. Representation is deeply dependent on the cultural environment and the community that uses the WIS. Intentions can be supported by providing stimuli that rouse or incite activity.

Intentions may be restricted by a *scope*. The scope allows to concentrate on the main aspects. Typical restrictions are the cultural environment, education or other profile properties of potential users, or specific time facets for utilisation of the WIS.

The first two facets of intention have a general form (objective, object) and a more concrete form (aim, target). The purpose facet depends on the mission and the audience, whereas the intent facet depends on the tasks. Therefore, the first facet specifies the 'what', the second the 'how', and the third facet the 'when' of an intention. We may either concentrate on a more general specification or a more concrete one.

The different facets may be considered separately or altogether in a condensed form. The detailed consideration is necessary, whenever a fastidious audience requires sophisticated content. High demands come into consideration due to the content provided, the community that must be satisfied, the sophisticated functionality required for the WIS, or the high attention the WIS gains.

EXAMPLE 3.1. The purpose facet is often considered to cover 'soft intentions' or by 'business rules'. In an information service a user may be interested or becoming more interested in some content. The intent facet is different and mainly driven by tasks the user tries to solve.

Similarly, we may derive a number of intentions a user has in mind whenever s/he visits an edutainment site:

Aim: An aim of an edutainment user may be to obtain a certificate that proofs the success of learning. Aims of providers of edutainment sites might also be binding the learner to some products such as the software of a supporter of the website. Typical general aims within an edutainment site are to grant equal rights to everybody; nobody should have higher rights than other learners.

Objectives: Typical objective of using an edutainment site are greater ease of learning, greater pleasure or satisfaction during leraning, and more fun. Another general objective is security and privacy. Edutainment applications also host confidential information, e.g. information on progress and errors. The learners need to be sure of the security and privacy of their data. They should be informed of the security precautions.

The *audience* of an edutainment site may be rather unspecific or more specific such as students of a college or analysts of a bank.

The *mission* of an edutainment site is to support learning by providing easy-to-grasp knowledge. At the same time, the provider may be interested in increased visitor-to-customer conversion rate, increased number of returning customers, and increased revenue. ◊

The intent is related to the more concrete tasks a user has in mind while visiting a WIS.

EXAMPLE 3.2. Intents come directly from analysing the needs and the demands of users. Some possible *tasks* a user may want to accomplish in an edutainment site are the following:

- A user realises that certain knowledge or information is necessary for solving a task. For instance, an analyst of a bank wants to know what the changes are in the customer community that led to a rise of faulty credits.
- A student in a college needs to know methods for analysing data. In this case, the student is interested in a data mining course that provides material on his/her educational level, permits learning the content interactively, and is comparable to the material the student is currently working with in the college.

Possible intents in an edutainment site are the following:

Objects: There are a number of objects such as faster task completion, successful completion of more tasks, or commission of fewer errors.

Targets: In the bank analyst case, the analyst needs to know data mining methods, to capture the achievements and the disadvantages and pitfalls of such methods.

The *themes* in the bank analyst case could be the interest in learning association methods, preprocessing of data, and prediction methods. ◊

The time facet represents general time restrictions such as the time or the interval for visits of the WIS, the time and the interval of repeated visits, or the temporality that may force a user to leave the site.

EXAMPLE 3.3. Visitors of a learning site can be characterised by their time dimension, i.e. the time they access and use the site, the access pattern they prefer, and the response time they request from the site.

Design: An edutainment site may allow a visitor to use the material on one shot or to use it step by step. In the latter case, didactics for the elaboration of material becomes an issue of WIS development. Step-by-step learning depends on the cultural environment of the user, the collaboration with other learners, and the interactive facilities of the site.
Furthermore, edutainment sites may be used over a longer time period. Therefore, context-sensitive help becomes important, because some actions or scenes may not be obvious to all users. Making such help available, e.g. through pop-up windows ensures that the learners do not lose time and interest.

End: Each action of a user leads to some effect that is either made visible to the user, recorded, or changes the state of the learning process itself. The ends of actions must be checked against the intended effect. This kind of consistency check is supported by acceptance conditions used for explicit checkout from scenes.

For edutainment sites, typical *occasions* may be the sudden requirement to explore some knowledge or to prepare a piece of homework. Another occasion might be the sudden demand in quality information due to some news popping up somewhere else. ◊

The representation facet represents the flavour or atmosphere of a site as discussed in [28]. This facet is largely determined by the other three facets.

EXAMPLE 3.4. An edutainment site that supports bank analysts should strive for an aesthetic and minimalist design. The main target for the representation is to support work. The site needs to ensure a clean and understandable layout. All irrelevant features must be omitted. At the same time an informative feedback is required. The site needs to be consistent throughout. The analysts want to control their way of working. At the same time they become experts of the site and thus need shortcuts and accelerators. Furthermore, they are restricted in their attention to the site, so they need a facility to recover from errors.

An edutainment site oriented to support pupils may follow completely different design principles. Pupils need to be attracted by graphical elements they like, they face in every days life, and which make using the site a pleasing experience. Pupils have a less-trained short-term memory. So, the short-term memory load is reduced, if pupils can recognize what they need to know from visible elements of the site. ◊

The set of intentions we should consider may be rather large. Moreover, the faceted representation may become too difficult to manage and to satisfy. The order we may use depends on the audience and on the provider. The audience is oriented towards solving tasks. The provider has a 'mission'. We can use these two restrictions to figure out which kind of website supports this profile, to harmonize our understanding with the corporate identity, to order the target realisation (short-term, long-term), and finally to develop an understanding how the site will look like in two years from now on.

EXAMPLE 3.5. Aims and targets can be ordered. We can use the mission and goals of the provider and compare them with the tasks the provider wants to support for different audiences. For instance, we may range the priority using a scale from 1 (highest) to 5 (lowest).

aims and targets	community customer	casual customer	reseller	hunter/ gatherer	kids	students
selling books	1	1	5	1	4	3
informing on books	1	1	5	1	4	5
reviewing books	1	1	5	1	5	5
selling bestsellers	3	1	4	1	2	5
developing a community	1	5	5	1	4	2

For business or information sites we obtain the following order:

1. primary intentions of the provider;
2. primary necessities and demands of the customers;
3. secondary aims of the provider;
4. stimuli for customers to revisit the system and to buy more;
5. audience education for the customer with our customer community;
6. beauty for customers;
7. customers self-realisation. ◊

The specification of the intended audience of the WIS is based on a weighted list of kinds of users, their aims, a comparison of the interaction with users we have so far, and a specification of aims of the audience and reasons for re-visiting.

EXAMPLE 3.6. Let us consider an information service that supports traveling, e.g. based on the brand

$$(\text{City,Hotel})^{\text{information}}2(\text{Tourist,Investor})^{\text{inform,book,invest}}.$$

This pattern is based on the following coarse specification of intention:

- *Purpose*: attraction of visitors and building up customer relations based on a *mission* that is centered around area information and attractions.
- *Intent*: travel convenience, hotel room sales and more depending on the *theme* travel support.
- *Occasion*: utilisation of links from the site or other associated sites.
- *Metaphors* used for presentation: survey and collecting basket.

The intention reflects tasks such as travel preparation, supporting visitors of an area, and general information for citizens and tourists. ◊

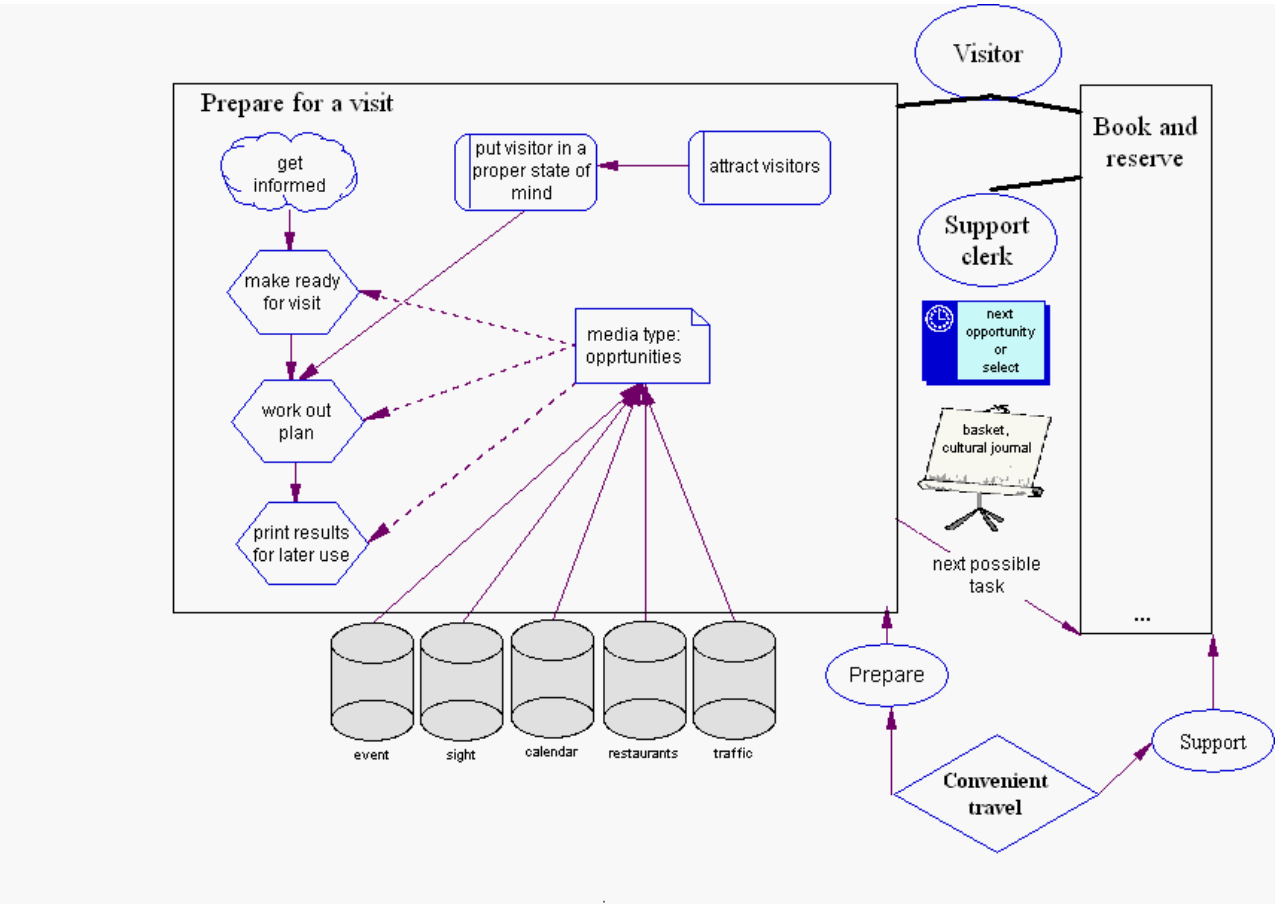


Fig. 3.1. The graphical representation for the intention *prepare for a visit*

EXAMPLE 3.7. The intention *prepare for a visit* is based on a number of intents such as addressing visitors beforehand for some purpose, use, or activity. It includes the aim to put the visitor of the WIS in a proper state of mind. A task associated with preparation may be to work out the details of a plan in advance. The task may be extended by putting pieces of information together into a compound form or preparing a report. The time facet may range from 'now' until 'next opportunity'. Typical metaphors supporting this intention are baskets, descriptions, and cultural journals.

The intention *prepare for a cinema visit* extends the intention of *prepare for a visit* by a certain content, a refinement of the time facet to the next possible visit, and a refinement of the purpose facet now targeting at entertainment.

We may use a graphical language for the representation of intentions. The language we are using extends the task-goal graphs considered in [28]. Tasks are represented by rectangles. Goals are represented by diamonds. We add now purposes, intents, data types that may be used for task completion, and the time and representation facets. Since purposes and intents refine tasks, we may place those into the tasks. Purposes, e.g. aims are represented by rounded rectangles. For objectives we use clouds. Resources, i.e. media types are represented by document icons. Intents are represented by hexagons. In addition, we may add the time facet, the representation facet, and supporting databases. Figure 3.1 displays a graphical presentation of the intention *prepare for a visit*. We have added links that show the dependence among the facets. ◊

We can combine the description of intentions in a semi-formal way as follows. The items in the list are optional except the first one.

`Intention space:`	⟨intention name⟩
`Purpose:`	⟨outcome description⟩
`Aims:`	⟨list of aims⟩
`Objectives:`	⟨list of objectives⟩
`Intents:`	⟨outcome description⟩
`Targets:`	⟨list of weighted targets⟩
`Objects:`	⟨list of weighted objects⟩
`Themes:`	⟨class of intents⟩
`Time:`	⟨outcome description⟩
`Design:`	⟨general flow⟩
`End:`	⟨effects, termination conditions⟩
`Occasion:`	⟨list of objectives⟩
`Presentation:`	⟨general style guide⟩
`Atmosphere:`	⟨general description of atmosphere⟩
`Metaphors:`	⟨list of metaphors⟩
`Based On:`	⟨*tasks*, *audience*, *mission*, *goal*⟩

Chapter 4

Life Cases

Life cases allow to overcome the information overload and lost in the hyperspace syndromes typically observed for WISs. For completion of tasks users need the right kind of data at the right moment of time, in the right dose, of the right form, in the complete extent, and within the restrictions agreed upon in advance. Moreover, users are limited in their abilities for verbalisation and digestion of data, and by their habits, practices, and cultural environment.

These limitations may cause intellectual overburdening of users. Most systems that require sophisticated learning courses for their exploration and utilization did not consider these limitations and did not cope with real life situations. The approach we use for avoiding overload is based on observation of real applications before developing the system.

We may extract life cases from observations in reality, which are very useful source, whenever a WIS is going to be developed from scratch or is required to provide a 'natural' behaviour. In the latter case users are not required to learn the behaviour of the WIS. Instead, the user can continue using a 'classical' behavioural pattern.

EXAMPLE 4.1. As a motivating example let us consider the life case *relocation* of a person, which consists of

- the change of basic relocation data including the possible removal of data on the old location,
- the change of official documents such as the passport,
- the optional change of relation enhancements such as the registration of pets, relocation of cars,
- the change of personal specific data such as family enhancements, or relationships to religious bodies,
- the change of data for additional relocation announcements such as tax, insurance changes, and
- specific additional tasks such as applications for housing allowances.

The person acts in the role of an *issuer*. We observe that *relocation* is enhanced by the profile of the issuer, by the specific tasks related to the *relocation* of the issuer, by specific laws and regulations, and by advanced functionality required for associating the life case with other life cases.

The life case *relocation* consists of steps such as *change of address data*, *change of data for associated people*, *change of registration data* for cars, pets, etc., *change of specific data*, e.g. data for public authority responsible for aliens, *change of data for social aid*, etc. These steps are bundled together due to their relationship to one person and to one life case. The associations may be represented by adhesion of different steps, e.g. representing the association of steps by a hypergraph, e.g. the one in Figure 4.1. ◊

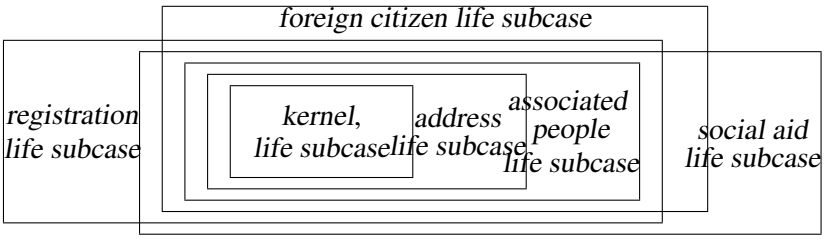


Fig. 4.1. Hierarchically ordered life case steps for *relocation*

4.1 The Concept of Life Case

Life cases are characterized by:

Observations: We are interested in the collection and assessment of behaviour relating to the specific application. This would typically involve an observation of behaviour of users in real environments, including a background check that relates usage to intentions, goals or tasks.

Processes: This involves arranging all the actions observed in an application into a main logical and coherent pattern. In some case, deviations of the main pattern must be modelled through exceptions. In most cases, we can use parallel execution of independent actions.

Assessment: This involves the reconstruction of the sequence of actions and specific behaviour of users. This will aid in understanding the role each individual has within the story. It assists in developing the user profile.

Individual profiles: A list of background including physical, and behavioural characteristics of individuals is conducted. This list can also be used for deriving the most appropriate interview technique we discuss below.

Interpersonal coherence: A variation in the activity will relate to variations of other life cases.

Significance of time and place: The choices made also depend on mobility, surrounding, and schedules of events of interest.

Characteristics of the life case: Individuals using a service may be grouped by characteristics. Based on this grouping a similar behaviour is observed.

Experience and skills: Individuals may have their own experience with services provided in real life and thus use different behavioural pattern of service employment.

In general, life case studies are designed to produce a picture of service employment that is as accurate as possible. Determining *why, how, where, when* and *why* a service is called using *what* data provides a better picture for utilisation scenario. As life cases are used for quality control, we must carefully record our observations. We either use a natural language specification or a semi-formal one as described later.

EXAMPLE 4.2. Let us consider the support of hotel search within an information service. In this case we may observe the behaviour of individuals in travel agencies while seeking for hotels. We observe that in most cases search based on associations is preferred over search by main properties such as name, address or facilities. Associations may be of a large variety, e.g. convenience to reach a hotel, location in certain maps, places of interest, or events that have caused the search. Other search criteria may be bargain or bundled offers. At the same time we observe that hotel search is combined with other intentions of users such as visiting cultural institutions. It may depend on results for search of other individuals.

Another example of a life case study is the booking of train tickets depending on individuals, offers of railway companies, circumstances of individuals are using trains, etc. ◊

4.2 Life Case Development

Life cases may be developed and interpreted step by step:

1. The first step during life case collection is the survey of possible application cases we might consider. The observations could have more than one meaning and may follow a variety of user-related goals. In this case we consider the most likely meaning.
2. The second step involves a deep assessment of the life cases. We extract the different execution orders, the relationship among actions, and the roles individuals play these actions.
3. The third step extracts those life case characteristics that are distinguishing features and are relevant for time and location. At the same time we search for similarities within these life cases.
4. The final step is concerned with the characterization of potential users, their behavioural patterns, and their roles in various stages of the life case.

Collectively, this information will produce a picture of the life case we are intending to support by a WIS. This may produce further life cases, or may aid in reducing the amount of development. It may result in a prioritisation of life cases under consideration, assist in the linkage of potentially related life cases, assist in assessing the potential of the WIS development, provide the WIS developers with relevant leads and strategies, and keep the WIS development on track and undistracted. These life case are mapped to scenarios in the sequel. The integration of scenarios can also be based on life cases.

EXAMPLE 4.3. Let us consider an information service for a city or a region, which supports a number of life cases:

- Attracting visitors: The information we are providing is based on some information need visitors may have. The life case follows those that we observe during marketing activities.
- Inhabitants information: The life case is based on selected information chunks that may be of interest to individuals, an ordering of the corresponding available information, and a derivable personal newspaper.
- Informing tourists: The life case is similar to those observed in city information centres.
- Providing official city information to inhabitants: This life case follows the message board metaphor that is used for newspaper information.

During the development of city information services such as the service `www.cottbus.de` we have made a life case analysis for city information services and detected around 30 different life cases. We can categorize these life cases by the content and information demand of individuals. In this case we distinguish:

- life cases related to tourist content for inhabitants permanently living in the city, for inhabitants temporarily living in the city, for short-term tourists and business people on short term trip, for vacationists planning a trip for more than 5 days, for teenagers with uncertain wishes, for seniors with particular interests, for week-end visitors, for visitors of highlights, festivals, etc.;
- life cases related to official content for inhabitants on search through directory of public authority, for inhabitants on search for emergency cases, for inhabitants orienting themselves before applying at public offices, for inhabitants interested in problems of city management, etc.;
- life cases related to business content, e.g. for investors considering engagement in the area.

For the collection and development of life cases it is normally a good idea to interview the personnel currently providing the service. ◊

Life case should be checked against sufficy. As they may have a variety of traces, we add two stages to life case analysis:

Exploration: The actual life case is provided to WIS customers and incorporated into their processes. Life case exploration can be conducted in normal environments of the user such as home or work. Then users can show the actions while they are explaining their aims and targets.

Apprehension: Finally, cross checking of life cases and utilisation scenario is used for correction, extension and affirmation of the developed utilisation scenarios.

During life case elicitation and specification users are confronted with sketches of scenarios. We ask what users think about these conceptualisations. The feedback is used for the refinement of life cases. We may use affinity diagramming, in which case we arrange the individual conceptions we discover on a blackboard, associate them, and discuss their relationship to the general characteristics of WISs. Another technique is based on card sorting similar to the model-view-control cards used in software engineering. In this case cards represent simple life situations. Their associations are then used for organising the conceptions. These sketches can also be used for discussing deviations from the normal case and for search of exceptional life cases. Instead of directing users to specific life cases we can show them two or more alternatives for the problem solution.

Life case analysis goes beyond task analysis. It is simple and easy to carry out, lightweight and straightforward to understand, and yet quite accurate in identifying the real demands users of a WIS might have. While observing real life cases and mapping them to life cases that must be supported by the WIS we carry out a user-centered development. Additionally, we may identify and resolve conflicting or competing intentions. These conflicts can be resolved on the basis of life cases by accommodating both intentions, i.e. fulfilling the stakeholder intentions and supporting the demand of users.

Besides observation of real life situations life case detection can also been based on *interview techniques*. Research on artificial intelligence has also resulted in a number of interview techniques:

- Unstructured interviews can be considered as an informal and explorative conversation on goals a user is following and on tasks a user has to complete. Unstructured interviews observe the rules that are applied to brainstorming. All interruptions should be avoided. The questions asked must be short and simple to answer and should be open-ended questions. Interview partners should share their thoughts and experiences. There is no judgement, confrontation or condescension. Users should not be lead toward a certain scenario. Unstructured interviews should only be interrupted if clarification is required. They need consecutive feedback in order to give the interviewees the impression that the interviewer is listening. While the interviewees are talking, everything that seems to be important is written down and used for later questions.
- Structured interviews are based on query plans. These plans can be based on the general characteristics of WISs. We start with easy questions on data and main functions and continue with the life cases of their utilization. Context and intentions are more difficult to capture. Structured interviews are also based on features, which can be shown to users for a rating of their importance.
- Life cases can also be detected by observing and critically analysing products of competitors. Elicitation of life cases from existing solutions is based on specifications, results from interviews with business users, excerpts from documents and spreadsheets, analyzed messaging and transactions, knowledge on the solutions that are currently used, meta-data and context information on the utilization within the current framework, and third party information. As reasons for restrictions cannot be captured, the copying of already existing solutions can only be used in exceptional situations.

- Users may also use protocols of "loud thinking", in which case they are provided with a real-life problem of a kind that they deal with during their working life, and asked to solve it. They imagine that they are solving the problem presented to them. As they do so, they are required to describe each step and the reasons for doing what they do. The transcript of their verbal account is the protocol. In this case, they work and explain their current work. These protocols can be the basis for capturing a scenario. This interview technique should be combined with other interview techniques.
- There are other interview techniques that might be useful for life case elicitation such as the laddering or grid techniques. In these cases, a hierarchical structure of the application domain is formed by asking questions designed to move up, down, and across the hierarchy.

4.3 Semi-Formal Representation of Life Cases

Although it is sufficient for life cases to be stated in natural language, we may use a semi-formal representation instead using the following template:

`Life case:`	⟨life case name⟩
`Characterisation:`	⟨outcome description⟩
`Tasks:`	⟨list of user tasks⟩
`Problems:`	⟨list of problems⟩
`Background:`	⟨general characterisation⟩
`Objectives:`	⟨list of objectives⟩
`Life case flow:`	⟨general graphical description⟩
`Milestones:`	⟨graph of milestones⟩
`Content consumed:`	⟨consumed content items⟩
`Content produced:`	⟨produced content items⟩
`Themes:`	⟨class of intents⟩
`Figures:`	⟨actors list⟩
`Expected profile:`	⟨general profile description⟩
`Collaboration:`	⟨general collaboration description⟩
`Context:`	⟨general context description⟩
`Time:`	⟨temporality limitations⟩
`Place:`	⟨assignment of places⟩
`Legacy:`	⟨names of documents⟩
`WIS:`	⟨general WIS context⟩
`Representation:`	⟨general behavior⟩
`Approaches:`	⟨general description of approaches⟩

This template captures the following components of life cases:

Characterisation: Life cases are characterised on the basis of strategic issues, the problem statement, background and objectives for the life case, the methodology that is used for solving the life case, and by describing the basic modules that are used for solving the life case. The characterisation is harmonized with intentions, tasks and goals.

Life case flow: The life case flow combines the observations we made, the processes involved, and the data which are consumed or produced. The life case flow is mapped to a scenario.

Figures: We develop profiles, especially those of individuals, as part of the WIS utilization portfolio, and interpersonal coherence specifications.

Context: Time and location is explicitly described for life cases. The applicability of life cases is restricted by regulations, laws, and orders. These restrictions are seldom given in an explicit form. In addition, the context of life cases is given by the provider, the intended audience, the utilization history, and the availability of data due to the technical environment. We also use this information for the context specification.

Requirements: Life cases are restricted by habits, general approaches, good practices, and boundary conditions for their application. They presuppose experiences and skills of the users involved.

Note that life case we intent to support by a WIS can be completely different in real life. Sometimes we need a complete reorganisation of the business activities. In this case we should not map the real life case to a suite of associated scenarios, but rather envision a better organisation of the tasks and goals and then map these to a new envisioned hypothetical life case.

EXAMPLE 4.4. For illustration let us consider a typical life case that is currently based on mailing and delivery services. Such a life case can be often observed, when a number of institutions is involved in the actual life case and these institutions collaborate loosely using a number of default and exceptional cases. Often, the latter ones are not explicitly stated, but belong to the service context.

For instance, a fee applies to kindergarten or day nurseries. This fee depends on the income of the parent(s) or legal guardians, their social status, the regulations of the day nursery, and the support of the government. Additionally, exemption of fees may be applied for. So, the observed life case consists in the following steps:

- Parents or legal guardians are filling in an application provided by the day nursery. This application is mailed to the youth welfare department.
- The youth welfare department checks the application. They use data provided by other city officials. The result can be the refusal or acceptance of the application, or the request to extend the application by further data.
- In the last case a new application form is send back to the applicant(s) that will be the basis for calculation of the fees.
- The applicants are providing additional data, e.g. data about their income. This new application form is mailed to the youth welfare department.
- The youth welfare department calculates now the fees that are applicable in the (special) case.
- The decision of the youth welfare department is mailed back to the applicants.
- The applicants may accept the decision made or may file an objection against the decision. The objection is mailed again. It may cause another partial or full assessment.

This life case should not be mapped to a supported life case. Beside unreliability of mailing services a good number of exceptions can be observed. We may use another approach that has been implemented within the Cottbus university service system. Here the applicant becomes the owner of the application and as such all new data associated with the application are made visible to the applicant. Whenever new essential data are added to the application the applicant is notified about it. If a reaction is necessary then the notification is extended by a deadline tracking life case. The applicant opens a number of views to other actors with rights to see, to update or to extend data. Concurrently, the youth welfare department uses another document tracking system. This system is based on a blackboard approach, i.e. all documents which are currently under consideration are placed onto the blackboard and are ranked by their priority and deadlines. The document tracking system is supported by a work support system. This system contains all necessary regulations, a number of samples, and a data acquisition system that extracts, transforms, and loads data from other collaborating systems. The work support system provides also support for handling exceptional cases.

Thus, the new life case consists of an application delivery and tracking life case, a document handling and assessment life case, and a planning, scheduling and work support life case. These life cases are mapped to scenarios, which are furtheron combined into a single scenario. The three specific life cases become views on the combined one. Furthermore, we observe that the new life case is far more general than the previous one and could be the basis for a generic application processing WIS. ◊

Chapter 5

User Models

User modelling has changed the development to human-computer interfaces, and allows to tailor systems to the users, their needs, abilities, and preferences. User modelling is based on the specification of the *user profile* that addresses the characterization of the user, and the specification of the *user portfolio* that describes the user's tasks, involvement, and collaboration on the basis of the mission of the WIS.

5.1 User Profiles

In general, user profiles are specified through the *education profile* based on an insight into the knowledge, skills, and abilities of potential users, the *work profile* with a specification of the specific work approaches of users, and the *personality profile* representing the specific properties of users.

5.1.1 Education Profiles.

The *education profile* is characterized by properties obtained during education, training, and other educational activities, i.e. education, capabilities, and application knowledge.

The *education* is characterised by the technical and professional training a user got. Technical training emphasises the understanding and practical application of basic principles of science and mathematics. Professional training places major emphasis upon the theories, understanding, and principles in such fields as science, and engineering. It results in erudition, knowledge, and literacy.

Capabilities of the user for task solutions are based on the attainment of proficiency in skills such as understanding the problem area, reasoning capabilities on analogy, realizing variations of the problem solution, solving and handling problems, communication abilities, abilities for explaining results and solutions, and abilities for integration of partial problem solutions.

Application knowledge depends on the application type, the application domain, the application structuring by subjects, the direct structure, the description, the context and environment of the description, the parameters of the application, and application functions:

- The application type covers configuration, production, construction, design, interpretation, error analysis, repair by replacement, rebuilding, and construction, planning, supervision, monitoring, prediction, classification, recognition, instruction, training, consulting, coordination, control, judgement, and evaluation.
- The application domain can be as diverse as science, engineering, e.g. construction, computer engineering, mechanical engineering, telecommunication, energy, environmental, aviation, etc., institutions, government, business, e.g. accounting, administration, finance, production, personnel, procurement, distribution, development, design, material, research, planning, management, etc., military, or journalism.

- Application structuring by the subject refers to title, purpose, domain, evaluation, basic knowledge, and terminology.
- The direct structure captures kinds of associations, e.g. nets, clusters, relations, dimensions, hierarchies, etc., deep or narrow structuring with or without regularity, and internal structuring of the application.
- Application structuring by description addresses the representation with depth, width, level and dimension, solutions with size and complexity, kinds of handling, e.g. procedural, declarative, graphical, combined or mixed, and naming.
- The context of the description deals with associations to the user, applications, space and time, dependence and evaluation by the general context, reasons, purposes, and particularities.
- The environment of the description refers to the imension of the application in terms of search and solution space.
- Parameters of the application can be time, space and others.
- Application functions are as diverse as accumule, observe, form, structure, judge, weight, summarize, aggregate, adjust, prove, prefer, determine, estimate, predict, calculate, derive, transfer, control, check, plan, solve, treat, combine, analyse, decide, execute, and feedback.

5.1.2 Work Profiles.

The *work profile* is mainly characterized by the task solving knowledge and skills in the application area, i.e. task expertise and experience, system experience, and information and interaction profiles.

The *task expertise* describes the exact and partial knowledge of data, procedures, algorithms, functions, constraints, associations, frames, graphs, schedules, rules, calculi, examples, objects, diagrams, etc. Partial knowledge is characterised by vagueness expressed via uncertainty, incompleteness, inexactness, inconsistency, and heuristics, variants of representations using evidence, evaluation, probability, confirmation, certainty, belief, confidence, support, rough sets, etc., calculi, comments, constraints, etc.

The *task experience* identifies both positive experience, e.g. applicable knowledge, strategies, models and theories, and negative experience, e.g. development, support or knowledge deficits, cooperation and planning support reduction, capability gaps, lack of effectivity, insufficient competences, etc. In addition, the users' experience in coping with errors, self-organization, planning, and abstraction can be specified.

The system experience depends on the systems to be explored and used. It is limited by the user's abilities to cope with information delivered by a system and system-supported collaboration, and to work with systems in parallel to working without system support.

The information profile is based on the information needs of a user discussed below, i.e. the intentions in approaching the system, the amount of information a user can cope with, and the complexity of information a user can handle. The information profile can be modelled by database schemata that are extended by a user's preferred way of browsing through information.

The interaction profile of a user is determined by his frequency, intensity, and style of utilization of the WIS. Users might be experienced in working with several workplaces or with only one, e.g. a single browser. The interaction profile supports a flexible communication, cooperation, and coordination of a given user.

5.1.3 Personality Profiles.

The *personality profile* of a user characterizes his/her *general properties*, his/her preferences in dealing with the WIS, and his/her *polarity profile*, which addresses psychological properties of a user.

General properties of the user or a user group are the *status* in the enterprise, community etc., *formal properties* such as name, address, password, etc., the *context* for performing a task such as position, allocated tasks in development, distribution and application, task area and customer properties, the *psychological and sensory properties* capturing vision, hearing, motoric control, information processing, speed of information uptake, attitude and anxiety, the *background and personality factors* capturing intentions, motivation, expectations, emotions, etc., *training and education*, *behavioural patterns* such as acceptance of system properties and methods, *required guidance* by help and tutoring systems, and *user type*, i.e. whether is casual, user, knowledgeable or expert.

For characterization of *user preferences* we may concentrate on those that are important for WIS development such as preferences for input and output devices and dialogues. For input and output devices this captures the *handling of types* such as text, picture, graphics, audio and video and the required support for these, the *preference of specific forms*, e.g. character-, line-, window- or form-orientation with presentation preferences for colour, shape, size, font, format, contrast, etc., the *required guidance and help during input or output* capturing tutoring, explanations, alternatives, etc., *command preferences*, e.g. the command style, and the *understanding of the input and output tasks* depending on the level and capabilities.

Preferences for the design of dialogues capture *dialogue properties* such as the information load, flexibility, complexity, expressive power, initiated actions, consistency, feedback, etc., *dialogue forms and styles* such as conversation, question/answer form, input/act forms and various presentation styles, *dialogue structuring*, e.g. closed dialogues, open discussions, undirected querying, interrogation, etc., *dialogue control* on amount, quantity, relevance, representation, etc., and *dialogue support*, e.g. by tools.

5.1.4 Polarity Profiles.

We may draw special attention to the *polarity profile* that is a part of the personality profile. The polarity profile governs the development of layout and partially also playout. So, we may classify users according to the scale in the following table. Six different categories as by the table below may be used for the characterisation of users. These categories may vary and further refined using facets.

businesslike	-	lyrical	conventional	-	inventive
unemotional	-	soulful	common	-	extreme
rational	-	sensitive	coated	-	cool
considerate	-	sensual	reliable	-	exceptional
			conventional	-	bohemian
classical	-	modern	traditional	-	vanguard
discrete	-	noisy	old	-	young
ageless	-	modern	colorless	-	gaudy
calm	-	disquietingly	even-tempered	-	exciting
noncommittal	-	intrusive	familiar	-	uninformed
			spiritless	-	lively
tough	-	tender	rustic	-	cosmopolitan
harsh	-	mawkish	natural	-	artificial
geometric	-	bloomy	coltish	-	serious
firm	-	mellow	simple	-	complex
robust	-	delicate	difficult	-	easy
angled	-	round	graceful	-	gracile

These polarity properties may be represented by Kiviat graphs. For instance, Figure 5.1 is commonly known for characterising people from the Bavaria region in Germany. This self-perception may be used for the development of style-guides or patterns for layout of web pages.

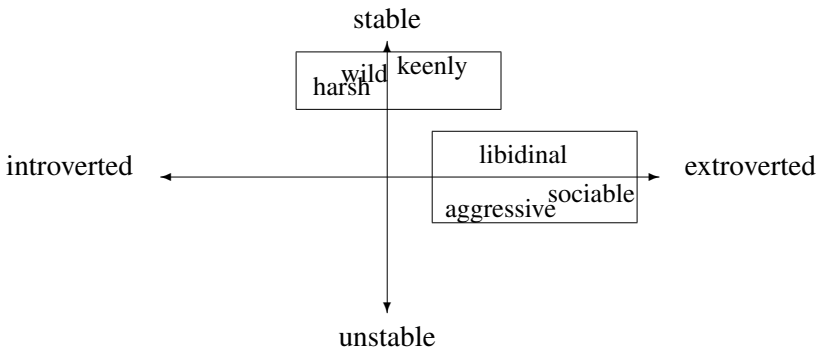


Fig. 5.1. The self-perception of the Bavarian

5.1.5 Representation of User Profiles.

User profiles can also be described in a semi-formal way as follows:

`User profile:`	⟨user profile name⟩
`Education profile:`	⟨general description⟩
`Education:`	⟨list of degrees⟩
`Capabilities:`	⟨list of skills⟩
`Knowledge:`	⟨description of knowledge in the application⟩
`Work profile:`	⟨general description⟩
`Task expertise:`	⟨description of knowledge⟩
`Task experience:`	⟨positive and negative experience⟩
`System experience:`	⟨experience with infrastructure planned⟩
`Information profile:`	⟨information need⟩
`Interaction profile:`	⟨interaction properties⟩
`Personality profile:`	⟨general description⟩
`General properties:`	⟨list of user properties⟩
`Preferences:`	⟨list of input/output/dialogue preferences⟩
`Polarity profile:`	⟨list of characteristics⟩
`Derivable profiles:`	⟨profile description and enforcement⟩
`Security profile:`	⟨access control and privacy⟩
`Safety profile:`	⟨safety requirements⟩
`Based On:`	⟨*user goals*⟩
`Based On:`	⟨*user types*⟩

5.2 Actor Profiles

We group users by their information demand and requested content, their utilisation patterns, and their specific utilisation and context. The abstraction of a group of users is called an *actor*. Actors are characterized by profiles and portfolios. Same as for users actor profiles consist of the *education*, *work* and *personality* profiles. Profiles are used for the derivation of preferences. These preferences are used for adaptation of scenes to specific actors. We look at the actor portfolios separately in the next subsection.

Actors may act in the role of an *end-user*, an *antagonist*, or an inactive *non-player character*. The three categories are commonly described by their properties, their abilities, subjects of their actions, and by changes they are imposing.

The profile is also used for the derivation of the *security profile* of the actors. It is specified on the basis of tasks to be performed by the actors, roles they play, and actions that are permitted or forbidden. The security profile includes all aspects of security, i.e.

- *access control* limiting access only to authorised individuals by applying policies for identification, authorization, and authentication,
- *privacy support* by preventing unauthorized parties from obtaining sensitive information, e.g. through support of anonymity and confidentiality.

This profile may be extended by the *safety profile*.

An actor takes part in some scenes of the storyboard. S/he is characterised by the character, location,actions, intention, and circumstances, i.e. what happened before s/he came there. He must discern the actions he is taking as well as the rhythms of the work as a whole, and he must determine what adjustments must be made in his/her storyboard for each of the other characters. Though actors are abstractions of user groups, we may characterize them by these abstract properties.

EXAMPLE 5.1. Let us consider an edutainment site. Based on the profile we can derive a number of preferences of learners. The background knowledge leads to different speed and reception of content by the learner. Work abilities and habits influence current work. Learning styles have many many facets depending on the profile of the learner. The social environment with cultural and psychological differences, the support for treatment of history of the learning process without annoying repetitions, and the the content change management with or without refresh are mapped to preferences we apply to the storyboard.

The profiles of actors can be specified using the following template:

`Actor profile:`	⟨actor profile name⟩
`Grouping criteria:`	⟨characteristics of grouping of users⟩
`Information demand:`	⟨general description⟩
`Utilisation pattern:`	⟨general description⟩
`Specific utilisation:`	⟨general description⟩
`Actor context:`	⟨general description⟩

Profiles can also be used for derivation of quality requirements of users.

EXAMPLE 5.2. For an e-commerce WIS we may derive a number of quality requirements depending on which kind of business the WIS is supporting. For instance, if the system is oriented towards selling after advertisement and presentation then actors that are acting as customers are interested in ease of use, speed, accessibility, previewing, support, community, safety, security, and guidance. If the WIS is supporting configuration sale, e.g. for travel, cars then customers are interested in customization, feedback, flexibility, previewing, background, contracts, safety, security, privacy, and guidance. If the WIS is supporting direct sale or reselling, e.g. auctions, first-in-first-out sale, or remnant sale then the quality critera that should be supported are ease of use, accessibility, preview, fun/community, alert, and privacy. ◊

5.3 Actor Portfolios

A portfolio is determined by the responsibilities one has and is based on a number of targets. The *actor portfolio* within an application is thus based on

- a set of tasks a actor has or intents to complete and for which solution the actor has the authority and control,
- a description of involvement within the task solution, and
- a collaboration that is necessary for solving the task.

Task modelling means to understand what a user wants to accomplish while visiting the WIS. At the same time, task analysis may lead to a reorganisation of the work processes to be supported. The WIS should support streamlining and augmenting what users do. Task analysis leads to a description of things users do and those they need to know. It does not specify how the task is accomplished. The tasks supported by a WIS need to be representative for the application, important within the application, and completely supported. Task support can be tailored depending on the profile and the context of the actors.

5.3.1 Tasks.

A *task* is a usually assigned piece of work, which often has to be finished within a certain time by a user or a set of users whose duty is its completion. It implies work imposed by a user in authority and obligation or responsibility to perform. A task may consist of subtasks, so we assume that tasks can be constructed on the basis of elementary tasks. Thus, a task is characterized by:

Problem statement: Tasks are associated to problems, for which often a class of solution strategies is provided. Additionally, problems often require collaboration with the local and global systems and with other actors.

Target states: After successfully completing a task we may observe a change in the state. Target states are specified by means of target conditions. Some of the target conditions can be optional. If no optional conditions are given then all conditions are obligatory. Target states correspond to intents.

Initial states: The necessity for tasks enactment is based on the insufficiency of the current state of affairs. Additionally, task enactment conditions may specify, under which circumstances we can start task execution.

Profile presupposed for task completion: The completion of a task requires skills, experience and knowledge that must be presupposed by the user, whenever the task is going to be activated. Tasks may be embedded into a certain organizational context. The profile also presupposes a certain technical environment, e.g. communication, information, and workspace systems.

Instruments for task completion: Task enactment is supported by instruments such as actions and data. Problems are solved on the basis of an information demand and within a class of functions that might be used for task solution. Leteron the information demand is mapped to database views or media objects. The function utilization is organized on the basis of workflows.

Collaboration profile: A task may be allocated to a single user or a group of collaborating users. In the latter case, the subtasks for each user, the obligations, the time and other restrictions must be given.

Auxiliary conditions: The settling of tasks may be restricted. Typical auxiliary conditions are based on rights for the direct handling and retrieval, roles of the antagonist and the protagonist, and obligations required whenever settling a task.

We extend tasks by a *general task context* that consists of the releasing actor(s), the actors to which task completion and results of completion are reported, the organisational unit that calls for task completion, the activities that are required by users, the aids that can be used for task completion, and the workspace that is supporting task completion. The latter is mapped to session support or to temporary workplaces of users.

Tasks are associated with intents a user or a group of users have. The intents are either objects or targets [39]. In the latter case, we can directly map the task specification to corresponding targets.

EXAMPLE 5.3. In a learning WIS for data mining learners have very different skills, very different background knowledge, very different approaches to learning, and want to use learning systems whenever this is really necessary. Let us consider tasks to understand the essentials of data mining and seeks for an algorithm that serves his needs.

1. The learner searches first algorithms that might be applicable within the application under consideration.
2. Next the user needs to understand the algorithm that seems to be the most appropriate. Understanding also includes learning the interpretation of results obtained by applying one of the algorithms to the learners data. For that, the learner may first look over demonstrations and illustrations for the chosen algorithm.
3. Now the learner experiments with the data. Data are selected and configured for the algorithm. The algorithm is executed and results are obtained.
4. Finally, the results are explained to the learner. S/he may now continue by selecting another method or algorithm and obtain additional insight into the data to be analyzed.

We may map these subtasks to the intents: exploring data mining algorithms, understanding how to prepare data, learning to interpret results obtained.

At the same time this task may be required by a life case [39]. For instance, the learner works as a clerk in a bank and tries to figure out which loans have become faulty and what is the reason for it. In this case, the clerk becomes a learner and tries to learn data mining as much as it is needed for the life case.

5.3.2 Task Execution.

The *task execution model* defines what, when, how, by whom, and with which data work has to be done:

- Task activities define how the work is actually done, e.g. by instructing the user or by invoking an application.
- The control flow defines the sequence of activity execution. Depending on the specific WIS several control constructs are available such as decision, alternative, parallelism, loops, start/exit conditions, and different kinds of constraints such as deadlines.
- The data flow specifies how data flows through the task. Depending on the WIS different concepts exist such as input/output-container of activities or local/global variables.

The *result of execution* is the state of affairs reached and the satisfaction of target conditions. Task enactment is supported by pre-determined plans. These plans are scenarios which have a number of stories they are supporting. We may use blackboards for recording open targets. Whenever a target is completed and a task is considered to be successfully fulfilled then we delete this task from the blackboard. The *involvement* of actors within the task solutions is based on the specification of the *role* an actor plays, the *part* the actor plays within the scenario, and the *rights* and *obligations* an actor has within the given role.

The role specifies the behaviour expected of an actor. A role is a comprehensive pattern of behaviour and serves as a strategy for coping with recurrent situations and dealing with the roles of others. A role remains relatively stable, even though different users occupy the position. A user may have a unique style of role execution, but this is exhibited within the boundaries of the expected

behaviour of the actor. Role expectations include both actions and qualities: for instance, in real life a teacher may be expected not only to deliver lectures, assign homework, and prepare examinations but also to be dedicated, concerned, honest, and responsible.

There are two types of roles: declarative and contextual ones. The former ones declare that an actor is playing a particular role, e,g, an actor being identified and as an employee. Contextual roles show how an actor acts within the context of a scenario and how an actor is involved within the context of another scenario. Declarative roles may be modelled by associating the actor to a role type. Contextual roles are modelled by associating an actor with the work effort the actor is assigned to and a role type describing the involvement of the actor. The role type provides a description of the role and can be hierarchically structured. Roles may also be hierarchically structured. At the same time roles may be played in collaboration.

EXAMPLE 5.4. In an e-business application we can distinguish roles such as worker, customer, and roles within the distribution scenario. The worker role entails roles such as contractor, contact, employee, and sponsor. The customer role takes part in bills to customer, ship to customer, and end-user customer. Within the distribution scenario we may distinguish active and pro-active roles, i.e. the distributor and an agent. These roles can again be distinguished into association, competitor, partner, prospect, referrer, regulatory agency, shareholder, supplier, and website visitor.

At the same time several roles are played by people or organizations within the context of project management. A person or organization may be the sponsor, worker, manager, lead, advisor or quality assurance manager for a project.

Users usually occupy several positions, which may or may not be compatible with one another. So actors may play several roles at the same time. In this case, they must have a combined view of the application and a combined access to their capabilities. Therefore, the WIS must provide the ability to *set-up* and *manage* these roles, and to *combine* portfolios so that the application maintains to be consistent for each actor.

The *part* associates the atmosphere, mission, objectives, and objects with the involvement of the actor in a scenario. It is refined, when context of the involvement is taken into consideration. The part is based on a categorization of the behavior of users. We may derive that users tackling some problem do not like to wait. The part is described through the kind of interaction actors are preferring for the current tasks and the behaviour of actors we need to support. Therefore, the behavioral stereotype is described through the part.

A role entails certain *obligations* and *rights*. Obligations are obligatory tasks, conduct, service, or functions that arise from the users role under specified conditions. Rights are granted as a peculiar benefit, advantage, or favor to a role. We specify right and obligations as conditions that are mapped to conditions on actions an actor can call in a scenario. Obligations and rights include such for functionality, for collaboration, for content, and for dissemination of such to other actors. Obligations and rights may include also the specification of obligations, permissions and prohibitions in the case of exceptional situations.

5.3.3 Collaboration.

Actors can collaborate for solving tasks. Such a *collaboration* is specified on the basis of the storyboard, the style of cooperation and the coordination facilities, and the roles of the partners, their responsibilities, their rights, and the protocols they may rely on. Collaboration is separated into *communication*, *coordination*, and *cooperation*. Communication can be based on classical architectures supporting communicating systems. Cooperation is based on an explicit handling of the work processes, the work organization, and on working spaces for collaborating partners. Cooperation can be

specified by storyboards. Coordination supports the consistency of work products, of work progress, and is supported by an explicitly specified contract that depends on intentions, on the users community and on the system supporting collaboration.

EXAMPLE 5.5. Collaborative learning is a teaching strategy in which small learning groups of learners, each of different levels of ability, use a variety of learning activities to improve their understanding of a subject. Each member of a learning group is responsible not only for learning what is taught, but also for helping group members to learn, thus creating an atmosphere of joint achievement. Learners may apply the knowledge they have obtained during learning steps. Each student may act in different groups in three different roles: the role of an exercise team member, the role of an advisor or expert and the role of an evaluator, who marks and grades the results of the team. The collaborating parties have their own cooperation, communication and coordination space. For instance, the evaluator and the teams share a group communication place and store their results in an answer delivery box. The evaluator delivers assignments to exercise teams through the assignment box. ◊

5.3.4 Actor Portfolios and Life Cases.

Portfolios refine the specification of life cases, which are collected by observing real life situations and the data and action flow in them. The life case description allows to deduct the demands for data, functions, performance, and supporting aids such as *workspace* and *workplace support*. The workspace is determined by the portfolio, its support by the profile of the actors involved.

EXAMPLE 5.6. Let us consider a life case *relocation*. The portfolio for this life case is based on a number of tasks such as: change basic data of issuer, provide a view on the data to all other interested parties, initiate tasks of associated life cases, archive old data and current changes, handle exceptional tasks, and change data in official documents. The life case is complex due to the large variety of subtasks, the initiation of which depends on the issuer. These tasks are associated with subtasks such as checking, whether all necessary documents have been supplied, special handling must be applied, etc.

At the same time we need to consider the involvement of at least four different actors. The issuer raises the change of data task handled by civil servants. If the issuer raises exceptions for data transfer then special support is required by an actor that acts as a data protection official. Finally, the data on the relocation are delivered to a number of actors that play the role of official bodies or support organizations. The security profiles of the issuer and the civil servants require a sophisticated data management, e.g. tax and social security data must be kept in isolation from each other.

The mindmap in Figure 5.2 summarises a part of the life case relocation, the tasks related to basic changes, to changes of associated parties, to change of data ownership, and to tasks of associated life cases. The actors participating in the life case are restricted to the four main types of actors. Their subtasks are different from the tasks of the issuer.

During analysis of user behaviour a number of principles needs to be observed. These principles form the *portfolio restrictions*.

Non-determinism of user behaviour: We need to cope with all possible options that are plausible. We do not assert that an option for execution will be followed. The various user intents are stored on a blackboard that is used to record tasks and subtasks to be completed.

Intention-based behaviour: We may assume that a user intermittently terminate actions as soon as their aims and their targets have been achieved. A typical implementation support for this principle is realized through the submit button. After submitting data or after hitting the submit button the user starts a new scenario or another scene under the assumption that the current intention has been achieved.

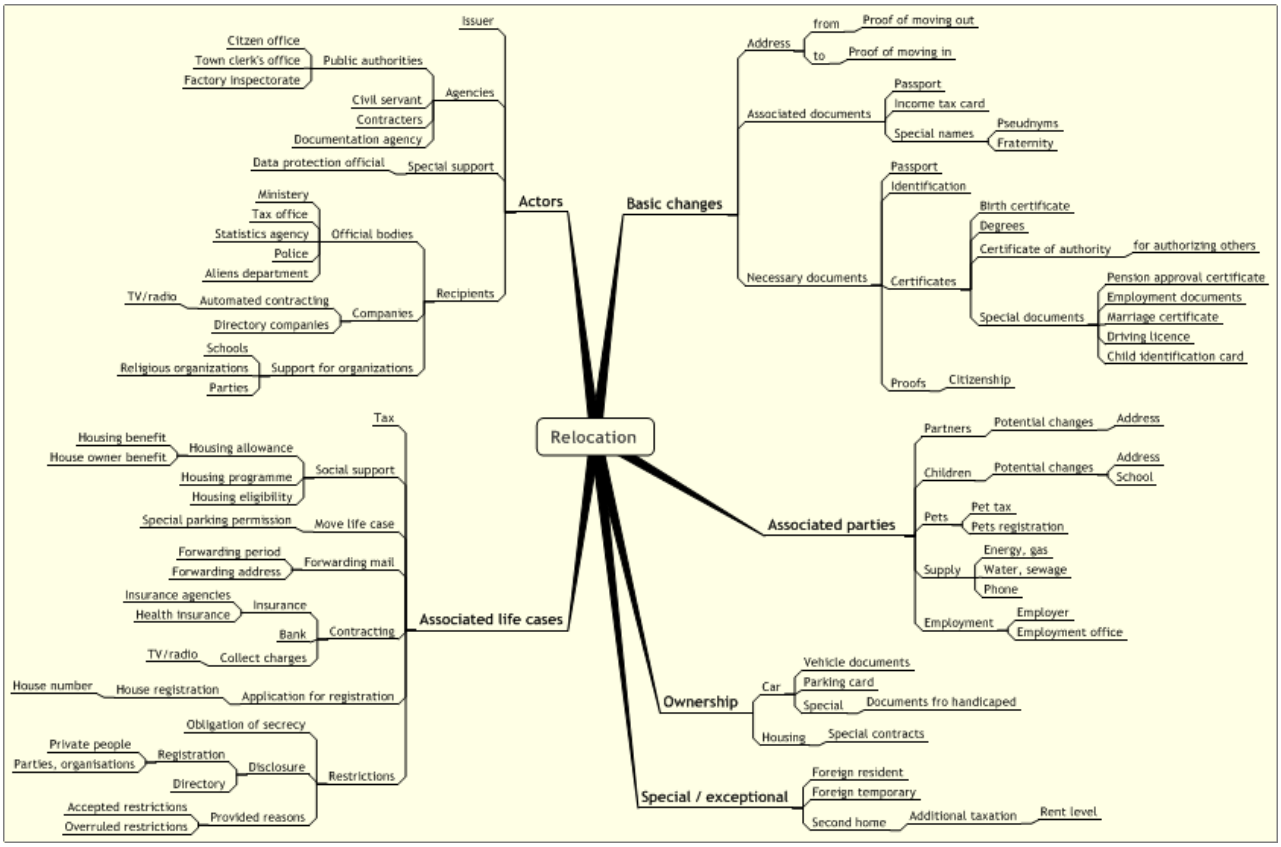


Fig. 5.2. The life case relocation with basic tasks, associated tasks, and actors

Task-oriented termination: We may assume that a user will terminate the current interaction when their whole task has been achieved. In achieving a target, subsidiary tasks might be generated. Termination is based on termination conditions. For instance, the user of an ATM machine expects that with termination the bank card is returned. Therefore, the user will have the card after termination of the ATM scenario.

Reactive behaviour: Users of a WIS are reacting to stimuli or messages of the WIS the user is observing. Rather than specifying each reaction in a separated form we use generic functions for the specification of this behaviour. Reactive behaviour can be modelled on the basis of observation-reaction pairs.

Collaboration target behaviour: The user enters an interaction with the knowledge of the task dependent on the targets that must be discharged. The collaboration target is based on a predetermined plan that is known in advance to the user. Once a target has been achieved the user does not expect to need to repeat the actions again.

Separation of mental and physical actions: Whenever the user commits to taking a physical action, then this action cannot be revoked after a certain point. Before doing the physical action the user generates the signal sent from the brain to the motoric system. We supply the user model by a guard-action pair linking mental actions with physical actions.

No-option-based completion: A user may abort a scenario when there is no apparent action the user can take that would help to complete the task. If e.g. the user of a railway ticketing system does not find the appropriate option, then the user might give up and might assume that the intentions cannot be achieved. We model this opportunity by a no-option-based addendum that is added to each scene where a user may leave the story.

Relevance: A user chooses an action only, if this action seems to be relevant for achieving the

currently considered targets.

These principles can be mapped to *control rules* of the storyboard. For instance, goal-based behaviour is mapped to acceptance conditions of scenes. The intentional non-determinism is reflected in non-deterministic scenarios. Task-oriented termination is mapped to invariants on stories, scenes and scenarios. Reactive behaviour is represented through the transitions between scenes. The user moves with an action from one scene to the next. To to so the action must be released by the system. We use collaboration targets for the derivation of actions that must be performed by the user. The necessity of actions can be modelled on the basis of deontic logics as described in the previous chapters. They are modelled through action guards. After a target has been achieved it is removed from the task blackboard. The guard-action pairs linking mental and physical actions are also used for refinement of the storyboard. For instance, if a user is not reacting within a certain time frame, then we may assume that the current guard-action is not valid and that the user needs action support or action feedback. The no-option-based addendum is added to each scene as a default addition. We may refine this addendum by actions that a user uses randomly after becoming confused. A possible refinement of the addendum is the incorporation of a wait function. Relevance is a quality criteria we apply during check of each scene. Only relevant actions may be fired by a user.

5.3.5 Representation of Portfolios.

Portfolios can be specified in a semi-formal way using the following template:

`Actor portfolio:`	⟨actor portfolio name⟩
`Task:`	⟨general description⟩
`  Extension of:`	⟨General characterisation of tasks⟩
`  Characterisation:`	⟨general description⟩
`    Initial state:`	⟨characterisation of the initial state⟩
`    Target state:`	⟨characterisation of the target state⟩
`    Profile:`	⟨profile presupposed for solution⟩
`    Instruments:`	⟨list of instruments for solution⟩
`    Collaboration:`	⟨specification of collaboration required⟩
`    Auxiliary:`	⟨list of auxiliary conditions⟩
`  Execution:`	⟨list of activities, control, data⟩
`  Result:`	⟨final state, satisfied target conditions⟩
`Actors involvement:`	⟨general description⟩
`  Role:`	⟨description of role⟩
`  Part:`	⟨behavioural categories and stereotypes⟩
`Collaboration:`	⟨general description⟩
`  Communication:`	⟨protocols and exchange⟩
`  Coordination:`	⟨contracts and enforcement⟩
`  Cooperation:`	⟨flow of work⟩
`Restrictions:`	⟨general description⟩
`  Actor restrictions:`	⟨general description⟩
`  Environment:`	⟨general description⟩
`Based On:`	⟨*life cases*⟩
`Based On:`	⟨*intentions*⟩
`Based On:`	⟨*general tasks, audience, mission, goal*⟩

5.3.6 Portfolio Context.

By explicit specification of the *portfolio context* we have a chance to cope with "errors" a user can make. Typical examples are the unnecessaryrepetition of an action, jumping back by the using the back button of the browser, reversing the order of actions, omitting or delaying actions, replacing one action by another, executing an additional action that is not required or unrelated to the current targets. As it is infeasible to support users in avoiding all these "errors", we specify the context together with the portfolio.

The context is used for the development of production rules for WIS implementation. A typical list used in some of our projects consists of:

Rules for blackboard control: The scenario is completed whenever the blackboard of tasks becomes empty. The scene data may be logged or removed.

Rules for information on actions: Actions that are not initiated by the user but are of interest to him/her need to be reported to the user.

Rules for control flow reporting: Users need a clear indication, if the order of actions that can be initiated by the user is not arbitrary. For example, a common practice is the utilization of a 'next' button. If the user does not know what that means then s/he is left in confusion.

Rules for control flow within scenarios: Although a user can choose among several options for actions we need to ensure that options are narrowed down to those that lead to meaningful stories. So, options that are available to the user are reduced at run-time.

5.3.7 Refinement of Life Cases.

Profiles and portfolios are used for the refinement of life cases. So far, life cases are specified by a general characterisation, the life case flow, figures, context, and requirements. With the availability of the user model, i.e. the profiles and portfolio of users and actors, we may extend the life case specification by:

Tasks associated with the life case: Life cases are normally not raised in isolation, but are associated with tasks that correspond to problems an actor intends to solve. These tasks have their own dependencies. So, we can associate with a life case other corresponding life cases. This association is mainly used for an extension of the life case under consideration.

Involvement of other actors and collaboration with them: The life case flow is refined by the cooperation specification of all collaborating actors. The involvement includes roles, rights, and obligations. Each actors plays a certain part that is restricted by time, data, processes, and specific restrictions of the portfolio.

Restrictions due to the profiles of participating actors: Due to the profiles of actors life cases are extended in order to support their specific requirements. The personality profiles are used for the generation of requirements according to the atmosphere of the WIS. Security profiles restrict the involvement of actors.

Context specialisation: The general context specified for a life case can be adapted according to the portfolio and the profiles of participating actors.

EXAMPLE 5.7. The life case *relocation* can be refined using a portfolio that is specific for the state Schleswig-Holstein or for a city such as Cottbus. If there are associated life cases or special cases, then another set of tasks is raised and the issuer is send to a number of offices. The city information system of Cottbus supports such life cases, e.g. the one depicted in Figure 5.2. In this case, the life case *relocation* is extended by all its associated life cases. In Cottbus one agency is handling all tasks

for the issuer. So, the issuer uses the collaboration with the city inhabitants agency. In the background this agency collaborates with all other agencies and recipients.

The life case *relocation* can be raised by any citizen. So, the profiles are rather general. We cannot expect that an issuer has a specific education profile, but we may expect that the issuer has a personality profile. This can be used for the life case flow, e.g. for checking first the existence of documents that are necessary and then raising the basic changes. ◊

This refinement of life cases is reflected in an extension of the life case template:

`Life case:`	⟨life case name⟩
`Tasks:`	⟨list of tasks associated with the life case⟩
`Actors involvement:`	⟨description of actors involvement⟩
`Profile restrictions:`	⟨list of restrictions due to actors profile⟩
`Context specialisation:`	⟨context embedding⟩

Refinement of life cases permits to derive requirements for the WIS development. These requirements support the designer by providing an input information to sketch the detailed content structure, the interface frames and grids, and the main functionality. Content developers in turn can define the main content demand needed for the WIS. The information we gain will impact on the content the WIS is going to provide, the structuring of this content, the access paths to content, the navigation, the presentation, the user operation, the WIS operation, and the interaction. Furthermore, these life cases may be used as a rich source for usability evaluation. We may also use these refinements for discussing non-functional requirements, which are usually expressed on the basis of user expectations.

The actor profile and the actor portfolio are also used for the derivation of requirements to the WIS.

Workspace: Actors need their own workspace in order to complete their tasks. This workspace may be provided temporarily, can be partially archived, and can be used for other tasks in the portfolio of the actor. Depending on the involvement of actors who collaborate to solve a task the workspace can also be accessible to other actors.

Session support: Actors are supported by session-defined media objects, i.e. certain content and functionality as defined later in part III. These media objects are generated whenever tasks are started, and may be adapted to the profile of the actors.

Collaboration infrastructure: Collaboration is based on communication, coordination, and cooperation of involved actors. In order to support collaboration an infrastructure is required that is extended or shrinked when new tasks are started or tasks are completed.

EXAMPLE 5.8. The workspaces for the life case *relocation* depicted in Figure 5.2 are different for the actors. The issuer does not need any workspace. The agency actors have a workspace in which a number of tasks may be bundled. Their workspace can be organized as a blackboard. The actor supporting data protection may use a completely different workspace that is based on monitor techniques. The issuer is supported by session objects that persist until all tasks of the life case are completed. Using this session support the issuer can resume the tasks at any time, e.g. by providing new data. The actors of the agencies collaborate in accordance with the legal restrictions and regulations. ◊

Chapter 6

Determination of Context

Taking the commonly accepted meaning a context characterises the situation in which a user finds him/herself at a certain time in a particular location. In this sense context is usually defined only statically referring to the content of a database. Only very few attempts have been made so far to consider context of scenarios or stories.

More generally, we consider context as everything that surrounds a utilisation situation of a WIS by a user and can throw light on its meaning. Therefore, context is characterised by interrelated conditions for the existence and occurrence of the utilisation situation such as the external environment, the internal state, location, time, history, etc. For WISs we need to handle the mental context that is based on the profile of the actor or user, the storyboard context that is based on the story leading to a situation, the data context that is based on the available data, the stakeholder context, and the collaboration context. These different kinds of contexts have an influence on the development of the storyboard and must thus be considered for the development of the WIS.

EXAMPLE 6.1. Let us consider a travel information system. It is often desirable to resolve the context of utterances. While booking an airline ticket to *London* the user may be asked for the airport code, for which s/he has a choice between LGW (London Gatwick), LHR (London Heatrow), LON (all London airports in UK), STN (London Stansted), and YXU (London, Ontario, Canada). The context of the travel request can be used to exclude the last option. The context of the airline used so far can be used to exclude two of the others. This context injection is based on the story environment and on content data. ◊

6.1 Context Space

When determining context we already know the major life cases we would like to support, the intentions associated with the WIS, the user and actor characterisation on the basis of profiles and portfolios, and the technical environment we are going to use. These restrictions enable a more refined understanding of context within a WIS.

The work in [28] characterises a WIS by six intertwined dimensions, one of which is context. We now relate context to the other dimensions, i.e. to the intentions, the usage, the content, the functionality, and the presentation. As presentation resides on a lower level of abstraction, it does not have an impact on context. Content and functionality will be used for context refinement, which we address later. So, we first concentrate on intention and usage. The user model, the specified set of life cases, and the intention can be used for a disambiguation of the meaning and an injection of context. In doing so we distinguish the following facets of context:

Actor context: The WIS is used by actors for a number of tasks in a variety of involvements and well understood collaboration. These actors impose their quality requirements on the WIS usage as described by their security and privacy profiles. They need additional auxiliary data and auxiliary functions. The variability of use is restricted by the actor's context, which covers the actor's specific tasks and specific data and function demand, and by chosen involvement, while the profile of actors imposes exceptions. The involvement and collaboration of actors is based on assumptions of social behaviour and restrictions due to organisational decisions. These assumptions and restrictions are components of the actor's context.

Storyboard context: The meaning of content and functionality to users depends on the stories, which are based on scenarios that reflect life cases and the portfolios of users or actors. According to the profile of these users a number of quality requirements such as privacy, security and availability must be satisfied. The actor's scenario context describes what the actor needs to understand in order to efficiently and effectively solve his/her tasks in the actual portfolio. The actor's determine the policy for following particular stories.

System context: The WIS is developed to support a number of intentions. The purposes and intents lead to a number of decisions on the WIS architecture, the technical environment, and the implementation. The WIS architecture has an impact on its utilisation, which often is only implicit and thus leads to not understandable systems behaviour. The technical environment restricts the user due to restrictions imposed by server, channel and client properties. Adaptation to the current environment is defined as context adaptation to the current channel, to the client infrastructure and to the server load. At the same time a number of legal decisions based on regulations, laws and business rules have been incorporated into the WIS.

Temporal context: The utilisation of a scene by an actor depends on his/her history of utilisation. Actors may interrupt and resume their activities at any moment of time. As they may not be interested in repeating all previous actions they have alrady successfully completed, the temporal context must be taken into account. Due to availability of content and functionality the current utilisation may lead to a different story within the same scenario.

We will discuss these various facets of context in more detail later in this section. This entire information forms the *context space*, which brings together the storyboard specification and the contextual information. Typical questions that are answered on the basis of the context space are:

- What content is required by the context space?
- What functionality is required by the context space?
- What has to be changed for the life cases, the storyboard, etc., if context is considered?

As outlined above the context space is determined by the actors, the scenarios, the WIS itself, and the time. It leads to a specialisation of the content, structuring and functionality of the scenes. We will discuss this specialisation in the last part of this chapter.

Context is associated with *desirable properties* of the WIS such as quality criteria and security and privacy requirements. Quality criteria such as suitability for the users or learnability provide obligations for the WIS development process. Though these criteria are rather fuzzy, they lead directly to a number of implementation obligations that must be fulfilled at later stages, i.e. within the development on the implementation layer.

For instance, learnability means comprehensibility, i.e. the WIS must be easy to use, remember, capture and forecast. This requires clarity of the visual representation, predictability, directness and intuitiveness. These properties allow the user to concentrate on the tasks. The workflows and the discourse structure correspond to the expectations of the users and do not lead to surprising situations. They can be based on metaphors and motives taken from the application domain. In the same way other quality criteria can also be mapped to development obligations.

Other properties that may be associated with context refer to the potential utilisation for other tasks outside the scope of the storyboard. In this case we do not integrate the additional tasks into the storyboard, but instead support these tasks, if this in accordance with our intentions. For instance, we might expect further visits targeting at core concerns of the WIS.

EXAMPLE 6.2. Sometimes customers may want to use a WIS for a purpose that does not meet the system's mission statement. For example, a customer may use a banking WIS to learn about the loan business, or a bookshop WIS to learn English. Clearly, the larger the gap between the actual customer's intention and the system's mission statement is, the higher the expected costs will be for supporting such customers. If it can be expected that some customers will interact with the WIS in a 'non-standard' way, a decision has to be made whether to support such intentions or not. This implies a modification of the anticipated information space. It shows that our focus on a business model for context modelling is not always a severe restriction. ◊

We may consider three additional context facets:

Provider context: Providers are characterised by their mission, intentions, and specific policies. Additionally, terms of business may be added. Vendors need to understand how to run the WIS economically. Typical parts of this context are intentions of the provider, themes of the website, mission or corporate identity of the site, and occasion and purpose of the visits of actors. Thus, providers may require additional content and functionality due to their mission and policy. They may apply their terms of business and may require a specific layout and playout.
Based on this information, the WIS is extended by provider-specific content and functionality. The storyboard may be altered according to the intentions of the provider, and life cases may be extended or partially supported. Provider-based changes to portfolios are typical for WISs in e-government and e-business applications.

Developer context: The WIS implementation depends on the capability of the developer. Typically we need to take into account the potential environment, e.g. hard- and software, communication channels, the information systems that are to be incorporated, especially the associated databases, and the programming environment developers use.

Organisational and social context: The organisation of task solutions is often already predetermined by the application domain. It follows organisational structures within the institutions involved. We captured a part of these structures already on the basis of the portfolio and modelled it by collaboration. The other pars form the organisational context. Collaboration of partners consists of communication, coordination, and cooperation. Cooperation is based on cooperativity, i.e. the disposition to act in a way that is best helpful for the collaboration partners, taking their intentions, tasks, interests and abilities into account. At the same time, collaboration is established in order to achieve a common goal. Actors choose their actions and organise them such that their chances of success are optimised with respect to the portfolio they are engaged in. Additionally, the social context may be taken into account, which consists of interactive and reactive pressures. Typical social enhancements are socially indicated situations such as welcome greetings, thanking, apologising, and farewell greetings.

6.2 Actor Context

Let us next take a deeper look into the facets of the context space, i.e. examining actor, storyboard, system and temporal context in more detail. The context of an actor is based on his/her intentions. According to the actor's profile s/he needs support to fulfil the expectations with respect to the quality of information and work. The social and intellectual interests of the actor may also be part of the

actor's context. The actor's profile may be used for a refinement of the actor's context leading to the following four specific kinds of context:

Actor projection context: Actors may act on their expectations. In this case, they intensionally drop portions of content or functionality and project the current content and functionality to the "normal" case. This projection leads to an implicit context. For instance, within a travel scenario actors are expected to behave like travellers. Another kind of projection is parameter suppression, in which case content or functionality may be dropped or is not noticed whenever it becomes partially irrelevant.

Actor approximation context: Often actors need first a condensed or approximated information that may be refined later. Typical such approximations are attribute value approximations or structural approximations. For instance, the former ones may allow the WIS to provide first an approximate value for the orientation of the user. A common misuse of approximation is pricing by "starting from". Structural approximation permits the use of the same symbol for the original object and an abstraction, hence enables the usage of simpler representations.

Actor ambiguity context: Sometimes the reference of a symbol can be unambiguous within a narrow scope, in which certain limitations apply, but ambiguous in a larger scope without the limitations. A typical unambiguous symbol is the 'next' button in case the next scene lies within the expectations of the actor. Another use of ambiguity can be made by choosing less expressive textual representations. For instance, in a loan application there is no need to clarify that the word 'bank' denotes a financial institution.

Actor mental context: The mental context captures attitudes and knowledge of actors or other kinds of alternative states of affairs such as fiction and user expectations. This context is described in terms of provenance, i.e. relating to real life cases or to expected life cases. Expectations of actors or users can be combined with other more general requirements. The knowledge of the mental context will remain highly incomplete. However, it provides a handle for incorporating users' and actors' expectations.

The actor context is intellectual as well as existential. It contributes to enabling the scenarios and the corresponding stories under consideration. The intellectual part is based on the profile of the actor, on habits, traditions, knowledge, experience, etc. It may be also based on the quality requirements an actor is imposing. The actor context restricts the users that might use the WIS, the way the system will be used, and the portfolios. It is based on the intentions for using the system and the portfolio of the actor, e.g. tasks, involvement, and the collaborations the actor is involved in. The existential part is also related to the portfolio under consideration. It is related to the data and functions currently available or provided, and the technical environment.

The specification of this specific actor context becomes necessary whenever we want to support the work of actors that is close to human communication. Human communication exploits the context often to an extreme degree, leaving many things implicit. We do not need a complete decontextualisation as long as the actor can interpret the content and functionality that is provided by the WIS. Contexts provide a mechanism by which we can use the simplest presentation, content and functionality, i.e. the ones that makes the fewest distinctions most of the time while transcending to more expressive presentation, content and functionality only when needed.

Due to these contextual abilities we may restrict presentation, content and functionality to those features that are absolutely necessary. These restrictions may result in presentation principles such as sparse utilisation of additional and not directly necessary content or functionality or economy in utilisation of colours, multimedia objects, and texture.

EXAMPLE 6.3. Let use Example 5.7 for an illustration of this principles. An issuer of the relocation life case expects that his personal and identification data are already sufficient for providing him/her

all necessary details. So, the context in which the issuer reacts is based on projection and ambiguity context. If we use the information the passport office provides as public information for the city office, then we can adapt the life case directly to the current one. At the same time, the visit of the issuer might be not the first such in his/her life. So, we can now use the information on previous life cases for scaling the life case to the expectations the issuer has.

The adaptation requires some background knowledge on the handling of life cases in other cities, previous visits, and the profile of the issuer. We may then use a number of questions to figure out, which further adaption or refinement of the life case is applicable. Since some data on the issuer cannot be stored in the system due to regulations and laws we need to repeatedly obtain these data. So, the data we need to capture within the life case are extended by data we need for figuring out which specific life case is under consideration. At the same time we may use this context information for adapting functionality that is provided. ◊

This specific actor context is combined with the portfolio restrictions. Actors with a non-deterministic behaviour do not use high ambiguity or deep projection. At the same time, their mental context and their approximation context must be rather sophisticated. Actors acting more on intention intensively use all four kinds of actor contexts. Task-oriented and reactive behaviour requires support for mental context. Actors acting in collaborations need additional support for their common disambiguation. If actors do not complete their tasks within one session, they need a well-prepared projection context for the case that they resume their tasks. We shall later map these requirements to adaptation rules and control rules for adaptation.

6.3 Storyboard Context

Context has also a storyboard dimension. The actor's context must be combined with the storyboard, life case and portfolio contexts. The latter two selectively condition the situational interest of the actor and the relevance of the current scene for the actor. Based on the relevance we may identify and use properly all the content that should exert the evolution of the current story. We may now use this information for extracting whether a sequence rule, i.e. a rule of the form $s_1 \Rightarrow s_2$ requiring that a visit of scene s_1 should be followed by a visit of scene s_2 can be applied to the current system usage. The rule may hold in general, but is considered to be not applicable if the existence of process p_1 leading to scene s_1 does not have a bearing on the existence of process p_2 leading to scene s_2. Therefore, the incorporation of context and the derivation of relevance has mainly to do with selecting the best story for the user and is thus used for the adaptation of content and functionality.

The storyboard context can be used for deriving the most appropriate content. We aim at delivering the right data to the right actor with the right tools and scope at the right time. As the storyboard context provides a good source for adaptation of content and functionality to the current stage of the scenario we collect context within the storyboard and add this context information to the context space. This context allows a treatment of the expectations of the actor. Therefore, each scene in a scenario is provided with a *pre-scene context*, *scene context*, and a *post-scene context*.

- The pre-scene context consists of all content that has already been delivered to the actor before the appearnace at the actual scene. This information can be used to reduce content delivery for scenes. At the same time, this content can be stored in condensed form and made available to the actor when needed, i.e. the actor can revisit the old content whenever this seems to be necessary. Classical browsers only provide a strictly sequential 'back' button for this kind of history management. The pre-context of a scene thus contains all valuable content that is collected during a story, and guarantees the availability of this content when needed.

- The post-scene context consists of a potential playout of scenes that can be entered after the current scene. If an actor needs some information on the next actions, then this context information can be used. This information is valuable for those actors who intend to drop out of the system. It is also a part of the help information. The post-scene context can be enriched by meta-data describing the content that is provided in the next steps or the data to be produced by the actor. In this case, an intelligent interface may forecast the information need for further steps of the storyboard.
- Each scene may also be enriched by superimposed meta-data on the scene, which include everything that could be referenced within the expected consumed and produced data. Typical such references are collaborating actors, retrieval or update data of the current content, and details taken from the log of the current story. Finally, the scene context may include administrative data such as identification of content currently under consideration.
  Scene context is enhanced by generic scene information, which can be based on intentions of the WIS. For instance, adverts may be attached to each of the scenes. Default information serves as an exception handling for scenes. If content or functions are currently not available, then default data are provided.

6.4 System Context

The system context is determined by the content and the functions that are provided by the web information system. It consists of at least the following four parts:

Source and acquisition: Source and acquisition is an orthogonal dimension of the WIS. A WIS is supported by media objects that belong to media types as we will explore in detail in part III. In a nutshell, a media object is defined by an extended view on some underlying database, which can then serve for provision of content and functionality of an elementary scene. The databases used for the generation of content form the context of the scenario. We may associate with each scenario the subschemata of these databases that are used for generation of consumed data or for integration of data produced by actors in the scenario.

Associated content: The data that are used for consumed and produced information do not exist in isolation. They are usually associated with other data on the basis of integrity constraints or existence constraints, in particular existence constraints are often not explicitly represented as such, but are embedded into the database schemata used. For instance, we usually associate with objects collected in relationship classes those objects collected in the component classes on which they are based. In this case, we assume that objects in component classes of the relationship type co-exist with objects in the relationship class. We need to consider the environment of content that is currently under consideration together with the data that are associated with this content.

Supported functionality: Functions supporting the actions in scenarios are provided by the WIS. These functions have their own control environment. Typical such control mechanisms are logging, concurrency control, and recovery management.

Security: Security concepts describe *encipherment* and *encryption* (keys, authentication, signatures, notarisation, routing control, access control, data integrity, and traffic padding) for data exchange.

6.5 Temporal Context

The temporal context appears in a number of variants, e.g. storage time, validity time, display time, user-defined time, transaction time, etc. The temporal context is applicable in a number of combinations. Sometimes it is necessary to use all of them, but often it is often observed that only one variant of this context is necessary.

Versions show the life cycle of the objects under consideration. As scenarios will have their own life cycle we cannot assume that database changes are directly enforced on websites. Moreover, it may be useful to provide the old content as long as an actor continues with the same story. Versions can often be systematically structured by *database system phases*:

- The *initialization phase* permits the development of objects storing initial information. Integrity constraints are applicable in a limited form.
- The *production phase* is the central phase, which consists of runtime querying, modification, transaction management, etc.
- The *maintenance phase* is used in productive database applications for clarification of soft constraints, maintenance of constraints that have been cut out from runtime maintenance, and changing the structuring and functionality of the entire database system. Maintenance phases are used in data warehouse applications for recharging the data warehouse with actual information.
- The *archiving phase* is used for archiving the content of a database in a form that data relevant for historical information can be easily retrieved. No data modification is permitted; the only modification operation is to load new changes to the archive.

6.6 Representation of Contexts

We may now combine this context information using the following semi-formal template:

`Context:`	⟨context name⟩
`  Extension of:`	⟨General context⟩
`Actor context:`	⟨general description⟩
`  Projection context:`	⟨expectations⟩
`  Approximation context:`	⟨condensations and abstractions⟩
`  Ambiguity context:`	⟨scope⟩
`  Mental state context:`	⟨general description⟩
`  Characterisation:`	⟨general description⟩
`Storyboard context:`	⟨general description⟩
`  Pre-scene context:`	⟨history of usage⟩
`  Post-scene context:`	⟨potential continuation⟩
`  Scene context:`	⟨superimposed meta-data⟩
`WIS context:`	⟨general description⟩
`  Source and acquisition:`	⟨system environment⟩
`  Associated content:`	⟨content environment⟩
`  Supported functionality:`	⟨function environment⟩
`  Security:`	⟨required security functionality⟩
`Temporal context:`	⟨general description⟩
`  Versioning:`	⟨general description⟩
`  Development phase:`	⟨general description⟩
`Provider context:`	⟨general description⟩
`Developer context:`	⟨general description⟩
`Organisational and social context:`	⟨general description⟩
`Based On:`	⟨*life cases, portfolio*⟩
`Based On:`	⟨*scenarios*⟩
`Based On:`	⟨*general tasks, audience*⟩
`Based On:`	⟨*mission, goals*⟩

6.7 Lifting Relations

Context evolves for actors, scenarios, systems, and over time. We model the relation between different contexts by *lifting relations*. Properties that are valid for a certain context may be lifted to another context. This transfer can be based on local model semantics.

For this recall that a context is determined by actor, storyboard, system and temporal contexts. So let $\mathcal{A}$ denote the set of actors, $\mathcal{S}$ the set of scenarios, $\mathcal{W}$ the set of system characteristics, and $\mathcal{T}$ the set of time units. Then we can take a subset $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{S} \times \mathcal{W} \times \mathcal{T}$ to represent a set of contexts. Furthermore, we use a family of contexts $\{C_i \in \mathcal{C} | i \in I\}$ and a family of statement sets (or theories) $\{\mathbb{T}_i | i \in I\}$ that are associated with these contexts. Of course, the theory $\mathbb{T}_i$ describes the properties of the context C_i.

On these grounds we may use local models $M_{i,j}$ for each of these statement sets assuming that the models we consider are enumerated by the second index. More precisely, the models $M_{i,j}$ determine the meaning of content drawn from a language $\mathcal{L}$ for describing content in view of context C_i. That is, we use a partial mapping $\Psi : \mathcal{L} \times \mathcal{C} \to \mathcal{M}$, where $\mathcal{M}$ denotes a set of pre-determined meanings for content in $\mathcal{L}$.

We may now distinguish the formula α occurring context C_i from the same formula occurring in another context by considering the context index i, i.e. we consider pairs (α, i). Lifting relations can be modelled by rules of the form

$$\frac{(\alpha_1, i_1) \ldots (\alpha_n, i_n)}{(\alpha, i)}\, \varphi$$

stating that the formulae $(\alpha_1, i_1) \ldots (\alpha_n, i_n)$ can be lifted to (α, i) under the side condition φ. In addition, a compatibility relation among local models is introduced similar to logics that capture possible world semantics. This compatibility relation is used for entailment and satisfiability. This approach allows us to reason locally and then to transfer the knowledge we gained to other contexts.

Based on this coarse clarification of basic notation we develop a number of facilities and extend the specification of the WIS:

Context space: The *content context space* is defined on the basis of the content $\mathcal{C}$, scenarios $\mathcal{S}$ and actors $\mathcal{A}$. In Example 6.1 we could use information on the travel and on the airline to exclude options that seem to be less likely. The content context space of a WIS for a given content-meaning pair $c, m)$ consists of precisely those contexts, under which the particular content will have that particular meaning, i.e.

$$\mathfrak{C}(c, m) \;=\; \{(a, s, w, t) \in \mathcal{C} \mid \Psi(c, (a, s, w, t)) = m\}.$$

Adaptation of content, functionality, and scenarios to the context that is currently available is based on *context infusion*. Applying transformation rules we change content, functions, and the presentation. Therefore, we use a context specification for the development of enforcement rules. These rules may restrict scenarios to more specific ones, extend or shrink content, and extend or remove functions.

Life case extension and specialisation: The general life case specification can often be specialised, if context is explicitly injected. We need both the more general life cases and the contextualised ones. Whenever the WIS is revised or extended, we can return to more general life cases and generate another contextualisation. Typical specialisations concern changes in the life case flow. We may specialise the data consumed by an actor in dependence of the actor's context. If we know that actors need special auxiliary information or conversely actors became more knowledgeable during the utilisation of the WIS, then we may adapt the data provided for consumption. At the

same time, we can specialise the figures according to the given context. In the same way spatial and temporal information provide a basis for refinement of life cases.
Life cases may be extended to requirements that were collected in the context space. The content context may require a more elaborated content to be provided. The supported functionality may require additional functions, content, or a specific presentation. Intentions may be more specific under consideration of context. For instance, if we want to support a certain usage of a WIS that was not originally intended but became important in order to maintain frequent visits, then the original life case is extended by those associated life cases.

Development of a context manager: Context is also bound to scenes and thus evolves within a story. We may expect that content enhances context. For this reason, we introduced the pre-scene context. Therefore, a subsystem that manages the context is needed. This context manager uses the lifting rules introduced above for transferring context to context for scenes, collaborating actors, and the WIS as such. The system also supports the rule-based development of logics over time. We cannot require that the rule system is complete, but it must be consistent. A useful property is commutativity, i.e. the results of firing rules does not depend on their order. The context management system enhances the dialogue management system by adapting and specialising the presentation and injecting context into it.

EXAMPLE 6.4. A typical context extension to functionality is associated with the problem to avoid that users trap into losing-track situations. Such situationd can be detected based on the user's behaviour, e.g. invoking the help function repeatedly on similar topics, repeatedly positioning on particular locations and performing similar operations on similar data, excessively navigating through information space without invoking any reasonable functionality, looking repeatedly for FAQs on similar topics, attempting to enter a discussion forum, and sending email to the site administrator.

User aid that can be provided for losing-track situations is giving access to a thesaurus of the subsystem the user is accessing. Furthermore, the respective business model may be exposed to the user together with an explanation that is adapted to a particular user type. Similarly, access to a FAQ list suitable for the user and the accessed subsystem may be given. Furthermore, improved search facilities and examples targeting at the subsystem accessed may be provided. ◊

6.8 Adaptivity

The idea of adaptivity is to equip the system with enough additional information and rules that would render it possible to engender the right content and functionality for the current situation. That is, the system is supposed to act according to the dictum 'you take care of the specification, and the system will take care of itself and adapt to the current use'.

Two content objects c_1, c_2 are *synonymous* in the context $C_i \in \mathcal{C}$ iff $\psi(c_1, C_i) = \psi(c_2, C_i)$. They are *totally synonymous* iff $\psi(c_1, C_i) = \psi(c_2, C_i)$ holds for all contexts $C_i \in \mathcal{C}$. They are *epistemically synonymous* within a scenario s for an actor a iff $\psi(c_1, C_i) = \psi(c_2, C_i)$ holds for all contexts $C_i \in \mathcal{C}$ associated with a and s.

Applications often require adaptation of processing context, e.g. to

- actual environments such as client, server, and current communication channel,
- user rights, roles, obligations, and prohibitions,
- content required for the portfolio of the current user,
- the actual user with his/her preferences,
- the level of task completion depending on the user, and
- the user's completion history.

Consider for instance e-learning or e-government websites discussed in [28] and [44]. Citizens may apply for a primary place of residence. In this case, their passport must be changed; otherwise, no change is required. Citizens with school-age children may have to complete additional documents. Completed documents may be decomposed into a suite of documents due to legal restrictions, e.g. by a data protection act requiring that data for city officials and service offices such as the unemployment agency must be separated.

Depending on the role of users, story completion may be scheduled sequentially for some users or in parallel for others. For instance, clerks in a city office may consider documents in parallel, while citizens complete their documents in a sequential mode.

EXAMPLE 6.5. Adaptivity may be required at run-time. For instance, people with foreign citizenship may be required to apply for a residence permit. Users may require a varying support depending on the environment that is used for the completion of documents. Users should be supported whenever they are interrupted during task completion. ◊

These requirements lead directly to the requirement to develop a facility for *mutable, adaptable scenarios* for different users, portfolios, and contexts. We shall return to this requirement after introducing templates in the next section. It is our target to develop generic scenarios that can be refined to scenarios by injecting context. This approach is more widely used for WISs than one would expect. For instance, almost all information sites of cities and regions provide a very similar hotel or event search. The reason is not the existence of a development monopoly but rather the evolution of these search facilities to semi-standards. These standards are not officially agreed, but have been formed by copying successful solutions.

Chapter 7

Using Life Cases, User Models, and Context for WIS Development

7.1 Architecture-Driven Development

Architectures of Web Information Systems

Already in the first phase of the development, a first rough sketch of a general systems architecture can be described. It contains rough sketches of components envisioned. The application domain description is refined in requirements prescriptions by quality criteria and by a general WIS architecture. This architecture is refined in the consecutive steps. It is based on a number of views:

The conceptual view consists of a rough architecture of the WIS and its components.

The application view shows the architecture of the WIS for the business user.

The technical view describes main technical components of the WIS, their interaction, their behaviour and their properties. Typically, the technical view is based on layering or multi-tier architectures of services with corresponding exchange frames.

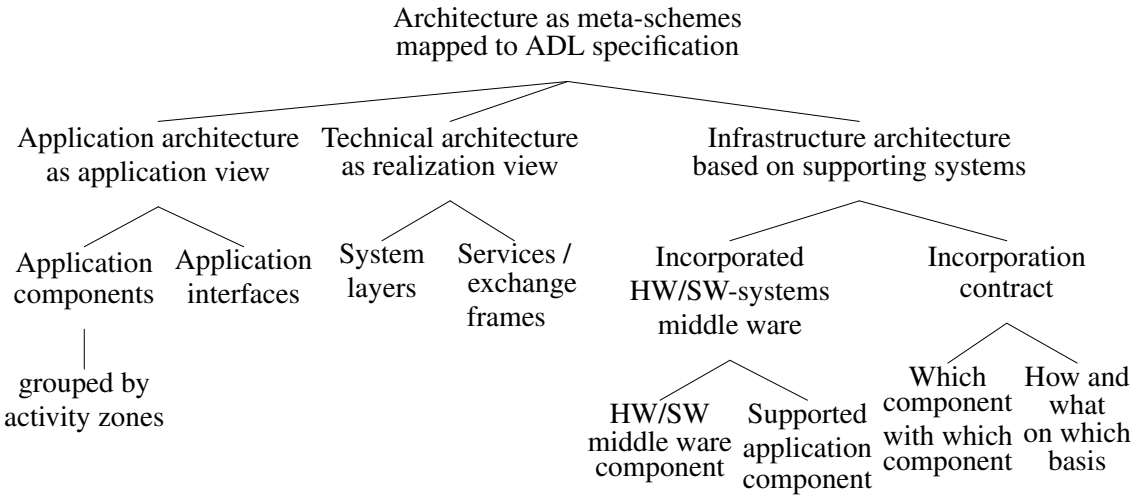
The infrastructure view displays the WIS environment, the middleware, the basic software, and the supporting software and hardware. Additionally, incorporation or deployment of components is described.

The execution view shows the activity map of the components for different stories. The execution view can be combined with the application view. In this case we use activity zones for depicting behaviour.

The module view describes the modules of the system. The program view displays the structure of modules and programs in detail.

We consider the WIS architecture to be harmonic if any view of the architecture can be displayed as an overlay view on the architecture.

Other approaches to architecture development are model driven architecture and model driven components. The following graph structures the main views we use for WIS-E.



The WIS architecture sketches the overall WIS by separating concerns, by clustering parts, by determining an overall component structuring and the corresponding interfaces and by separated treatment of each of the components. Architecturing aims in decomposition of requirements into subsystems. This decomposition results in component development and in definition of exchange frames between these components.

The application architecture is the abstraction on the architecture that shows the playout of the WIS depending on the story space. The quality criteria, their priority, and their analysis are combined in an *application architecture*. The WIS is shown from the viewpoint of the application. Constituents of the application architecture are survey on the components envisioned, their behaviour, services and interfaces. Components are abstractly specified in a black-box form. The inner structure is not provided.

The *technical architecture* provides a description of components of the WIS that are necessary for operating the WIS. Layering of services or presentation via multi-tiered architectures support also analysis, e.g., treatment of errors.

The *architecture of the technical infrastructure* is oriented towards a presentation of those hardware and software components and their communication models that are integrated into the WIS. Classically all peripheral devices, system software, specific protocols are prescribed.

Architectures for Component-Driven Development

Component-driven development is based on a choice of the architecture. We know a number of architecture styles:

The depot style is based on a central depot that contains all data and functions shared by the other systems. Compilers are often based on the depot style. The depot is generated during the first pass and consists of symbol tables, syntax trees etc.

The model-view-controller style (MVC) in Figure 7.1 separates different parts of the system: the computing system for the application domain, the view system for points of view on the system, and the controller system for interaction with the user and among system components.

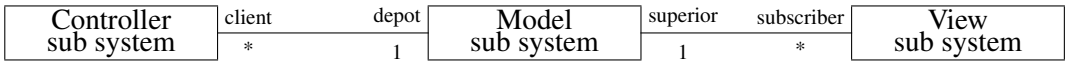


Fig. 7.1. The classical model-view-controller architecture

The client-server style is based on an exchange between the service calling client and the supporting server. Servers are providing services. Services are called by clients.

The P2P style (peer2peer) is a generalisation of the client-server style.

The n-tier style separates different aspects such as the presentation, the interface, the application logic, and the storage. Separation von Aspekten

The pipe-filter style uses sub systems as connected sequentialised I/O automata. Each of the automata produces an input for its succeeding neighbor and consumes the output of its preceding neighbor.

Interactivity of information systems has been mainly considered on the level of presentation systems by the Arch or Seeheim [15] separation between the application system and the presentation system. The interaction with the application system is based on a set of views which are defined on the database structure and are supported by some functionality. The web interaction style is additionally based on a standardised functionality of the client. The standardisation allows to use interfaces independently of the browser. Programming is based on scripting languages and device independent presentation forms. This principle has also been applied to text processors such as Tex processing systems.

The classical Seeheim model in Figure 7.2 has been used to separate the application engine from the user interface engine. Whenever the application engine is able to handle any user request, this architecture can be successfully used. When users require dynamic pages and the behavior of users is heavily dependent on the previous dialogue steps, the interaction has to be supported efficiently by the engine as well. Thus, we have to integrate structures, processes, semantics, and interfaces. Requirements for web information systems result in yet higher quality criteria for interfaces and their support by the engine.

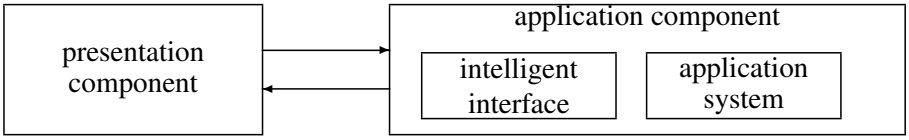


Fig. 7.2. The classical Seeheim 3-tier architecture

The Seeheim model is based on a separation of the user system and of the application system. We can thus observe a general exchange frame for messages and data exchange. The exchange frame in Figure 7.3 allows layering of systems. This observation directly leads to a proposal that generalises browser-based client-server architectures.

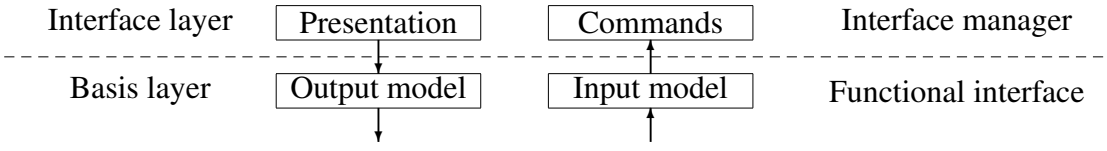


Fig. 7.3. The general exchange frame of interface management systems

We may generalise the Seeheim architecture to architectures that provide more comfort than thin clients which use browser technology. The architecture displayed in Figure 7.4 uses a 5-tier architecture.

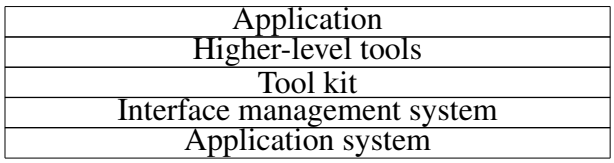


Fig. 7.4. Modern architecture of user interface software

These architectures are supported by *application programmer interfaces* (API). The Seeheim model uses an information exchange based on protocols. The flexibility is relatively high but the danger to reach a state that is not manageable is relatively high. The generalised architecture in Figure 7.4 also supports control of interfaces such as the command *kill*, supports the generation of adapted interfaces on the 'look and feel' paradigm, and the consistent management of active and passive interfaces. This management includes also selection of presentation styles such as *desktop*, *overlapping*, *tiling*).

The interface management system supports an association of the input stream of the user to the corresponding media objects ('klick-to-type') and the movement among interfaces ('move-to-type'). Such systems separate the data input stream, the control stream, and the computation stream.

We can combine the architecture of the Seeheim model and of interface management systems into the architecture displayed in Figure 7.5 This architecture has been proposed in [38].

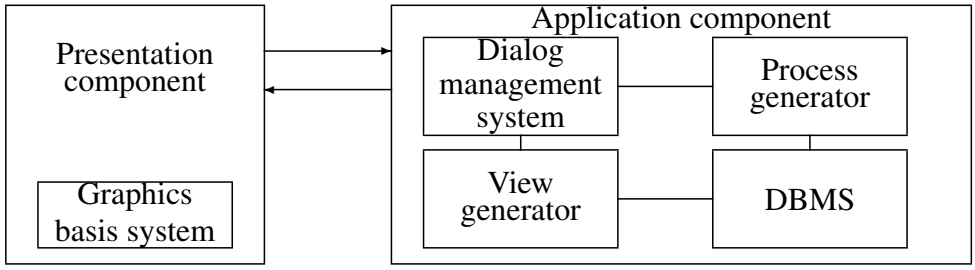


Fig. 7.5. The modern Seeheim model for WIS

Both architectures in Figures 7.4 and 7.5 use a separation into the presentation system. This separation leads directly to the separation of different concerns during WIS development: development of the information system and development of the presentation system.

Architecture-View Based Development

The proper specification of the WIS architecture supports scoping of the WIS development. We identify the following development views:

Refinement scope: determining those components that are affected by the intended refinement and specification of execution of sub-steps to be undertaken;

Data manipulation scope: determining the data consumption and data production of components that are active in scenes of the story space;

Function chunk scope: determining those components that are or become active during one collaboration step or function chunk execution with their active interfaces and the the corresponding data to be exchanged during the collaboration step;

Content chunk scope: determining those components that are necessary for processing the content chunk;

Scenario scope: determining those components that are used in scenes of the scenario with their interfaces;

Controller scope: determining those components with their used interfaces that are monitored by the controller or monitor;

Connection scope: determining those components and interfaces that are used by certain channels.

These development views are used for controlling the refinement of the work products and for analysis of the impact of development steps.

7.2 The Dichotomy of Systems

WIS have two different faces: the systems perspective and the user perspective. These perspectives are tightly related to each other. We consider the presentation system as an integral part of WIS. It satisfies all user requirements. It is based on real life cases. The dichotomy is displayed in Figure 7.6 where the right side represents the system perspective and the left side of the ladder represents the user perspective.

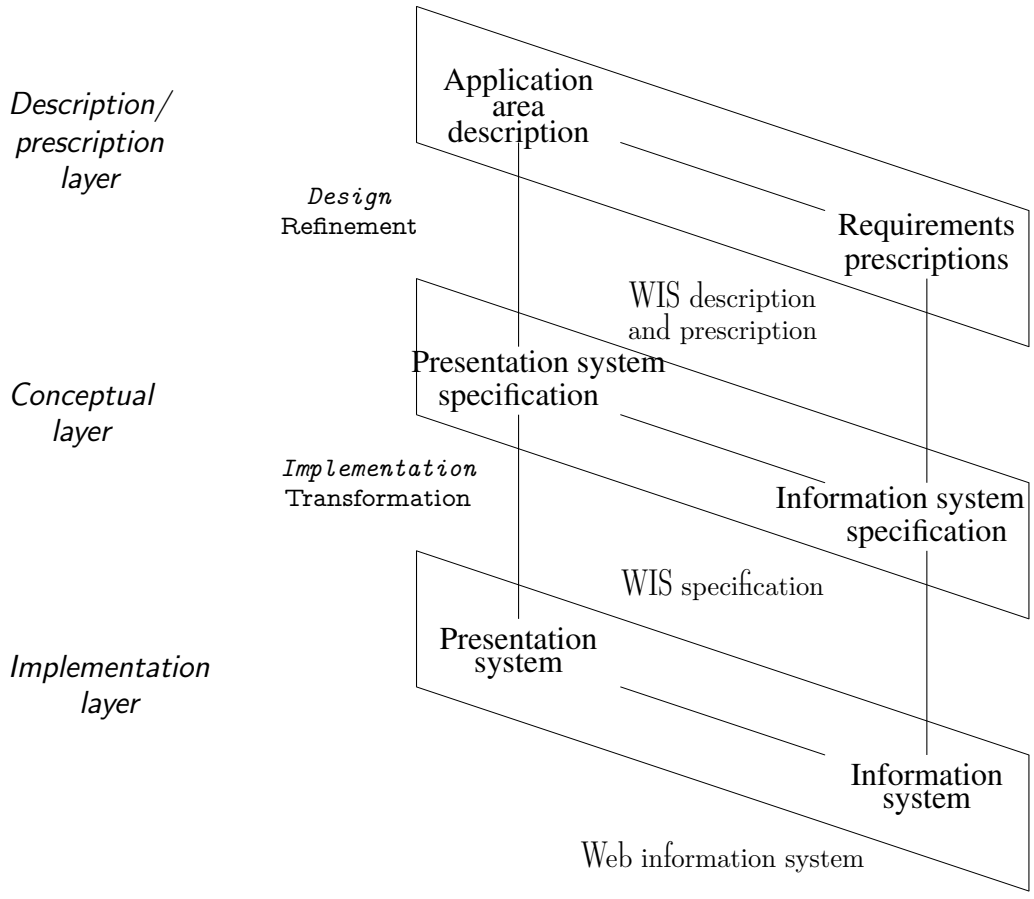


Fig. 7.6. The dichotomy of human-computer systems and the systems ladder

Software engineering has divided properties into functional and non-functional properties, restrictions and pseudo-properties. This separation can be understood as a separation into essential properties and non-essential ones. If we consider the dichotomy of a WIS then this separation leads to

a far more natural separation into information system requirements and presentation systems requirements. The system perspective considers properties such as performance, efficiency, maintainability, portability, and other classical functional requirements. Typical presentation system requirements are usability, reliability, and requirements oriented to high quality in use, e.g., effectiveness, productivity, safety, privacy, and satisfaction. Safety and security are also considered to be restrictions since they specify undesired behaviour of systems. Pseudo-properties are concerned with technological decisions such as language, middleware, operating system or are imposed by the user environment, the channel to be used, or the variety of client systems.

Properties are often difficult to specify and to check. We should concentrate on those and only those properties that can be shown to hold for the desired system. Since we are interested in proofing or checking the adherence of the system to the properties we need to define properties in such a way that tests or proofs can be formulated. They need to be adequate, i.e. cover what business users expect. At the same time, they need to be implementable. We also must be sure that they can be verified and validated.

Top-down development of systems seems to be the most appropriate whenever a system is developed from scratch or a system is extended. For this reason, we may differentiate among three layers: the systems description and prescription layer, the conceptual specification layer, and the systems layer. These layers may be extended by the the strategic layer that describes the general intention of the system, by the business user layer that describes how business users will see the system and by the logical layer that relates the conceptual layer to the systems layer by using the systems languages for programming and specification. Figure 7.6 relates the three layers of systems development.

The system ladder distinguishes at least between the following refinement layers: description/prescription, specification, and implementation. The refinement layers allow to concentrate on different aspects of concern. At the same time, refinement is based on refinement decisions which should be explicitly recorded. The implementation is the basis for the usage. The dichotomy distinguishes between the user world and the system world. They are related to each other through user interfaces. So, we can base WIS engineering on either the user world description, the systems prescription, the developers presentation specification, the developers systems specification. At the later deployment phase we distinguish between the user behaviour and the systems behaviour. The explicit distinction and the thoughtful integration and refinement of the different perspectives support a sophisticated WIS engineering.

We may extend the ladder by introduction layer, the deployment layer, and the maintenance layer. Since the last layers are often considered to be orthogonal to each other and we are mainly discussing WIS engineering the three layers are out of our scope.

The Classical Presentation System Development

Classical approaches to web information systems are often based on late integration of presentation systems into the WIS information system. This approach is depicted in in Figure 7.7.

Classically several layers of abstraction are identified. The top layer is called the *application domain layer*. It is used to describe the system in a general way: What are the intentions? Who are the expected users?

The next lower layer is called the *requirements prescription layer*, which is used to concretise the ideas gathered on the application domain layer. This means to get a clearer picture of the different kinds of users and their profiles. This may also include the different roles of users and tasks associated with these roles. The major part of this layer, however, deals with the description of the story board. Stories identify possible paths through the system and the information that is requested to enable

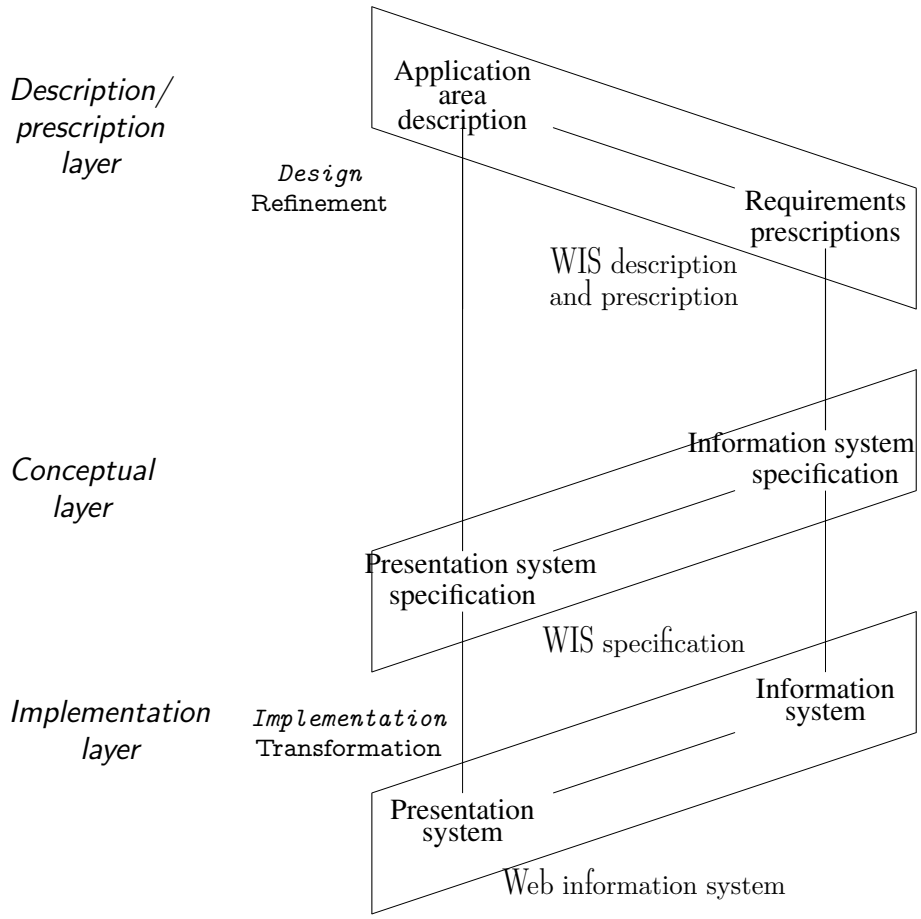


Fig. 7.7. The classical dichotomy of human-computer systems and the systems ladder

such paths. So the general purpose of the business layer is to anticipate the behaviour of the system's users in order to set up the system in a way that supports the users as much as possible.

The central layer is the *conceptual layer*. Whilst the requirements prescription layer did not pay much attention to technical issues, they come into play on the conceptual layer. The various scenes appearing in the story board have to be analysed and integrated, so that each scene can be supported by a unit combining some site content with some functionality. This will lead to designing abstract media types. The information content of the media types must be combined to design the structure of an underlying database.

The next lower layer is the *presentation layer* which is devoted to the problem of associating presentation options to the media types. This can be seen as a step towards implementing the system.

Finally, the lowest layer is the *implementation layer*. All the aspects of the physical implementation have to addressed on this layer. This includes setting up the logical and physical database schemata, the page layout, the realisation of functionality using scripting languages, etc. As far as possible, components on the implementation layer, especially web-pages, should be generated from the description on the higher layers.

This approach has the advantage that the presentation system specification is based on database views. The entire presentation depends on the maturity of the information systems specification. For this reason we may prefer the development according to the methodology depicted in Figure 7.6.

7.3 The WIS Specification Triptych

The software engineering triptych consists of three areas: description of the application domain, prescription of systems requirements, and specification of software systems. Within our approach we extend these areas by description, prescription, and specification of the presentation system. We may, furthermore, specialise to information systems. Therefore, the general development can be based on the ladder depicted in Figure 7.6.

Classically, requirements are based on a description of properties that a system shall maintain or offer. They are compiled into a documentation which prescribes desired properties of a machines, i.e., what the machine should and should not offer of data, control and functions. Since machines do not operate without an environment, environment description is often included into requirements.

Elicitation is typically a difficult task. The difficulty is caused by:

- Thin spread of application domain knowledge: It might be distributed across many resources and is rarely available in explicit form. There might be conflicts between different resources and partners due to intentions and portfolio.
- Tacit knowledge: The complete and consistent description of applications becomes harder if the application is in regular use.
- Limited observability: The partners might be too busy or too concentrated while using the existing system. The presence of an observer may change the behaviour.
- Bias: Partners may not be free or may not want to describe what should be described due to the political climate or organisational factors or due to the expected outcome.

We observe, however, that this task becomes feasible if application domain description is available.

WIS specification is oriented towards systems that are easy and intuitively to use. Therefore the classical information systems development model adds to systems development the development of presentation systems too. Classically user interface are built after the system has been developed. In this case, the user has to learn how the system behaves and must adapt his/her behaviour to the systems behaviour. We repair this mismatch by primarily considering the user worlds, the user stories, and the applications.

The main aim of WIS development is software that can and will be sold. Hence it must be a system whose use pleases people and which solves a problem. Therefore, the specification of the application domain is of primary importance.

Requirements must be reasonably well understood before software can be designed, programmed, and coded. The application domain must be reasonable understood in all its dependencies, weak and sufficient properties before requirements can be expressed properly.

Software engineering classically considers only the second and the third dimension. Application domain engineering is understood as the pragmatical side of the storyboarding approach. We, thus, concentrate our development on the first dimension: application domain description.

The application domain description is the first dimension of website development. Application domain engineering has not yet got its foundations. Our website development methodology includes this phase as the first phase of the development. Life case specification, intention description, profile and portfolio description, and sufficient understanding of context forms the main methods for application domain description. It describes the application domain as it is or as it should be without any reference to systems.

It contains descriptions of the phenomena that can be observed and descriptions of concepts, i.e. abstractions that these phenomena embody

- things,
- functionality,

- activities, and
- concepts.

The application domain is described by a number of properties, phenomena and various aspects that are desirable. A description includes designations, definitions, and refutable assertions. It can be formal or informal, sets a scope and a span, and expresses moods. Designations consist of a name, a recognition rule which purports to designate the phenomenon, and a general specification of the set of possible interpretations. Lexical definitions state the meaning of an expression in terms of other expressions. Ostensive definitions point to examples. Stipulative definitions assign a new meaning to an expression. Definitions may set bounds Refutable assertions are claims that may be shown to be true or to be wrong.

Application domain description can be based on the following major stages:

1. modelling business processing facets of a domain,
2. modelling intrinsic facets of a domain,
3. modelling possible support technology facets of a domain,
4. modelling possible management and organisation facets of a domain,
5. modelling possible rules and regulations of a domain,
6. modelling possible script (story) facets of a domain, and
7. modelling possible human behavior facets of a domain.

The life case specification together with the context description, and the user description which is based on the profiles of user combines these major stages. So can can use this description for an elegant and sophisticated elicitation of requirements and for the derivation of the presentation system specification.

The life case study combines cognitive techniques with contextual approaches. It is extended during WIS utilisation analysis by traditional approaches. This combination allows to avoid the known disadvantages of the more specific elicitation approaches.

7.4 Mapping Life Cases to Word Fields and Associators

The input for the application domain description are those properties in the reality which govern the application and exist independently of the WIS.

- User: The general description of the users characterises users, their intentions, their profiles, and their information demand. User intentions gather the reasons to visit the WIS.
- Context of the WIS in general: The WIS is typically supporting applications within a certain organisational framework. This context restricts the WIS.

As already discussed above, the application domain description consists of the collection of the input data and of the life cases. The description is usually extended by a description of the brand of the WIS. Figure 7.8 displays their dependence.

The application domain description results in a characterisation of the brand of the WIS, life cases with relative importance for the WIS, and of the context of the WIS depending on technology

The description is thus the input of the requirements prescription. During requirements development we use word fields as basis business activities (business use cases and events) and associators representing the general life case chart. We can also use life case to extract the portfolio and tasks that must be supported by the WIS. Users can be grouped according to their roles, rights, and obligations into actors. We can thus use user profiles and portfolio for development of actor profiles and portfolio.

The mindmap in Figure 5.2 can be decomposed into verb word fields and substantive word fields:

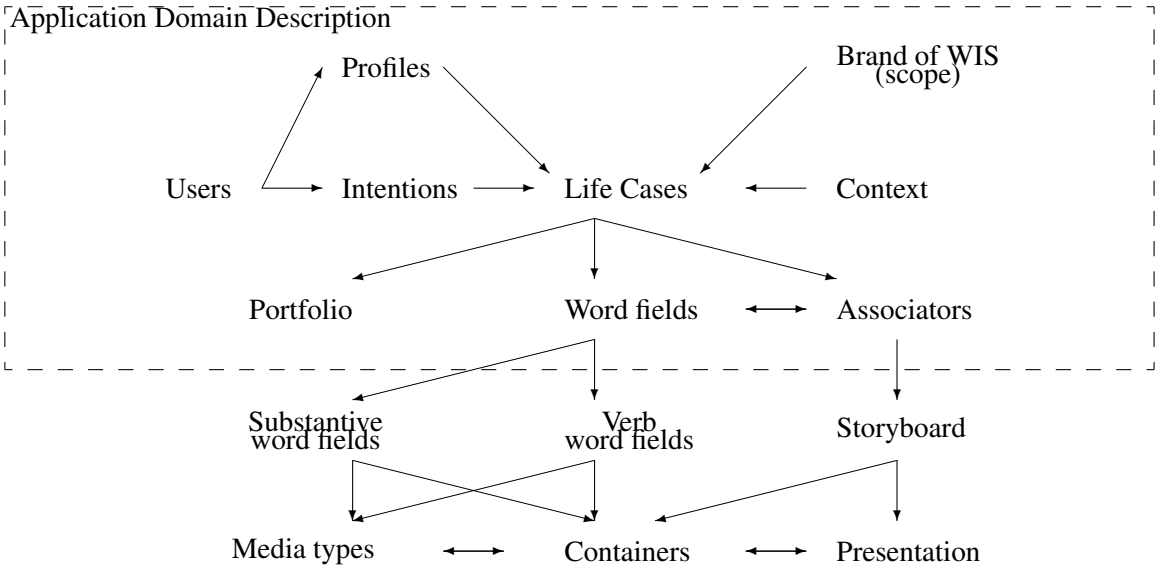


Fig. 7.8. The development results in relationship to input information to requirements

Governing verb word fields are the main verb word fields. They control the application, enforce other activities and bound roles.

Sign on: Issuers are signing on in an agency. This word field is the ruling the entire process. It also assigns the main roles such as issuer, supporter, recipient, and agency.

Sign off: The issuer cancels content or changes content to out-dated with the potential help of the supporter.

Register in specific form: An inhabitant might be a foreigner or might register a secondary place of life. In this case special actions are enacted or specific conditions apply.

Weakly bounded word fields relate to the life circumstances but are not tightly bounded.

Move: Whether an issuer really move, how the resettlement is performed and at what date the relocation happens is an issue that is related but not bound.

Report: The relocation data are, for instance, given to insurance companies and to companies contracting the issuer such as water and energy supply companies.

Redirect: The relocation data are used for redirecting mail etc.

Associated verb word fields depend on the content for the given life cases.

Re-register official documents: Documents which use the address data must be changed as well. For instance, the driving licence is going to be registered in the new location.

Apply for social support: Some issuers may be legible to apply for support. So their address data must also be changed.

Managing rights: Issuers may apply for specific rights such as parking in restricted areas. In the case of relocation rights may be cancelled and issued.

Governed verb word fields are those verb word fields which completion is requested by the governing verb word fields.

Relocated associated people: In the case of a family relocation the relocation procedure may also be applied to all people related to the issuer.

Official reporting: The change data are send to tax offices, to schools, to social services such as the pension service, and to the area offices.

Recording changes: All changes imposed are recorded for later proof of activities.

Re-register pets etc.: Issuers might have to register associated properties and pets.
Control: The issuer must provide a number of documents proving identity and proving legality of relocation.

Governing substantive word fields are the main substantive word fields. They bind content as consumed and produced content. Most European languages use subject words and a number of object word in the sentence structure.

Address data and all data associated with address data in official documents are subject to change with or without recording of the modification.
Issuers data are under change as long as they are related to the relocation life case.

The life case characterisation usually also provides an order and a hierarchy of the application of the word fields. We can use a number of associators

- Basic associators are sequence ; (execution of actions in sequence), parallel split $|^{\wedge}|$ (execute actions in parallel), exclusive choice $|^{\oplus}|$ (choose one execution path from many alternatives), synchronization $|^{sync}|$ (synchronize two parallel threads of execution by an synchronization condition ^{sync}, and simple merge $+$ (merge two alternative execution paths). The exclusive choice is considered to be the default parallel operation and is denoted by $||$.
- Structural associators are arbitrary cycles * (execute actions with or without any structural restriction on loops), arbitrary cycles $^+$ (execute actions with or without any structural restriction on loops but at least once), optional execution $[\,]$ (execute the action zero times or once), implicit termination $\downarrow$ (terminate if there is nothing to be done), entry action $\nearrow$ and termination action $\searrow$.

The associator language is similar to the action language we use for storyboarding [40, 41]. Therefore, it can easily be mapped to the storyboard.

7.5 Mapping Life Cases to Content Chunks, Function Chunks, and Storyboards

Requirements prescription contains a prescription of

- content chunks representing content, concepts and topics,
- function chunks representing all functionality requirements,
- the storyboard of the WIS,
- media type requirements (data, content, functions, containers, performance, operational, maintainability, support, security),
- representation requirements (look and feel, grids and pattern), and
- storyboard requirements (usability, humanity, privacy, cultural, political, legal).

The last three kinds of requirements are classically discussed for software engineering. The first two kinds are generalisations of the software engineering requirements to WIS. The third requirement is specific for WIS.

The information demand of all actors we would like to support is combined into content chunks. Additionally we can include also the specific information demand of users matching to the groups of users. This information demand can be combined into a singleton chunk of content that is demanded by all actors of similar steps in the life case. This information demand can be modelled within a database model.

Content chunks are mapped to media objects during WIS specification. The description of the data is given on the basis of media types, a specification of viewpoints, and a description of adaptation facilities. Content is also based on semantics. We thus associate content with concepts. Concepts are basic pieces of knowledge or their annotations. The usage of content may be restricted by preconditions. If content is used we might require that this content usage is recorded and generally restricted by postconditions. Content must also be annotated. Annotations are based on a commonly agreed vocabulary or dictionary or thesaurus. This agreement is bound to actors or to communities. We denote items of dictionaries or thesauri by topics. The pragmatics dimensions also reflects the context of potential usage. Since content is often used in a form that combines content chunks with other content chunks we also related content chunks to other content chunks

Content is thus provided by a *media object suite* which is essentially a complex media object. A suite consists of a set of elements, an integration or association schema and obligations requiring maintenance of the association. In the case of a media object suite, we specify media objects based on a type system enabling in describing structuring and functionality of media object, in describing their associations through relationship types and constraints. The functionality of media objects is specified by a retrieval expression, the maintenance policy and a set of functions supporting the utilization of the media object and the media object suite.

A function chunk consists of a set of functions which are bundled together by an association schema that is maintained by consistency requirements. For instance, the life case *relocation* consists of functions such as *change of address data*, *change of data for associated people*, *change of registration data* for cars, pets, etc., *change of specific data*, e.g. data for public authority responsible for aliens, *change of data for social aid* and so on. These functions are bundled together due to their relationship to one person and to one life case.

These functions are related to different views and different functionality provided. We use the theory of media types that combines views and their functions into one type. The media types may be iteratively defined on other media types. For instance, in our running example, we may distinguish input data for the storyboard, retrieval data for the storyboard, output data of the storyboard, display data suites for each scene of the storyboard, and escorting data supporting the understanding of each scene of the storyboard.

Function chunks used for operating content are mapped to media types. Function chunks may also result in functions for operating content chunks or for support of the storyboard. The latter are mapped to containers [40, 41].

The application domain description is mapped to requirements prescription. The corresponding mappings are given in Figure 7.9. Notice that content chunks represent subjective word fields and that function chunks represent verb word fields.

We thus can use mappings of the word fields and of associators for derivation of the requirements prescription:

- Life cases are decomposing to word fields and and synthesised through associators of these word fields. The separation into word fields allows the development of story scenes. The associators reflect the transition from one scene to the next scenes.
- Associators or rough stories are combined into storyboards. This combination is based on the separation of the life cases into word fields. The associators are reflecting the ordering, hierarchy and control within the storyboard.
- Word fields are mapped to chunks. These chunks represent data and/or functions. The data schema is based on the structure of the word fields and thus based on star schemata.
- Chunks are mapped to media types. These media types form a part of the conceptual schemata. They are based on views on the conceptual schemata to be developed during web information systems specification.

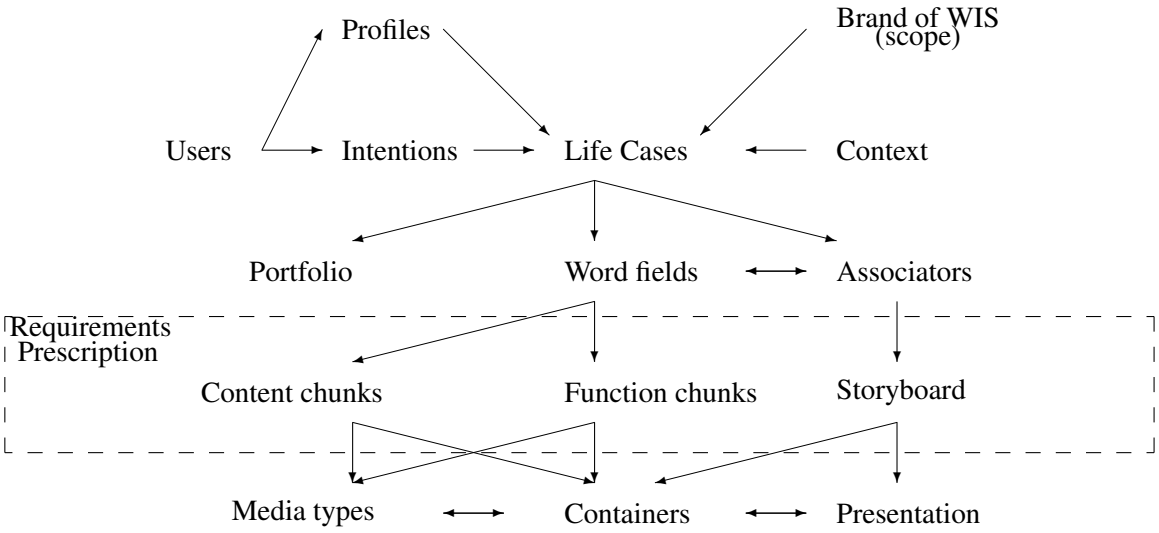


Fig. 7.9. The use of application domain information for requirements elicitation and analysis

- Chunks and storyboards are also used for the derivation and development of containers. Containers are used to deliver content and functionality to the user depending on the context. They may support an entire scene by corresponding functions for playing the role in the scene, by content that is consumed in the scene, by content update for data that are produced in the scene, and by providing an appropriate representation.
- Storyboard presentation is based on design patterns and grids. Each scene that is mapped to a web page is based on a presentation grid at the the conceptual or WIS specification layer. This grid is based on pattern which are prescribed at the requirements analysis layer. The pattern is described at the application domain description.

Chapter 8

Conclusion

In this paper we approached the pragmatics of Web Information Systems (WIS) design focusing on the method of storyboarding that is an integral part of the codesign approach to WIS design [40, 41]. A storyboard specifies in an abstract way who will be using the WIS, in which way, and for which goals. Thus, the specification of a storyboard captures the navigation paths, i.e. the stories through the "scenes" of the WIS, the action scheme accociated with the stories, the actors appearing in the scenes, and the tasks the actors accomplish. In addition, there are various static, dynamic and deontic constraints governing the storyboard.

While syntax and semantics of storyboarding have been well explored, its pragmatics has not. While many methods for WIS design emphasise content modelling, we start from the very fundamental observation grounded in semiotics that content refers to a syntactic dimension, whereas a pragmatic dimension requires dealing with information. This led to the objective to investigate in depth intentions associated with a WIS. The facets of intention arising from this form the basis for our technical development in this paper dealing with life cases, user models, and contexts.

Life cases capture observations in reality, which by means of abstraction can be used to derive scenarios for the storyboard. Integrating these scenarios provides a method for storyboarding. User models are by user and actor profiles, and actor portfolios. The latter ones provide a better understanding of the tasks associated with the WIS. Contexts can be classified according to how they impact on the life cases, the user models, and the storyboard extracted from them.

This work on pragmatics of storyboarding contributes to closing a gap in the codesign methodology for WIS design. It links the formalism of storyboarding to the systems requirements, and provides guidelines and means to derive the complex storyboards from informal ideas about a WIS without any technical bias. So, one one hand, this work on pragmatics is a decisive part of the methodology, which does not just consist of a collection of formally integrated models, but also has to state how to use them. It would be rather difficult mapping life cases or user models directly to a conceptual model of a WIS, which resides on a much lower level of abstraction as the storyboard. So on the other hand this work emphasises the need for storyboarding as the decisive tool for high-level WIS engineering. As shown in [43] this is also the basis for high-level reasoning about WISs addressing such important issues as personalisation of functionality.

Despite the high relevance of pragmatics for the completeness of storyboarding and the codesign methodology as a whole, the work reported in this paper is only half of the story, as it only addresses the usage analysis. We are in the process of writing up a second part of storyboarding pragmatics dealing with WIS portfolios, which combines content and utilisation portfolios that give rise to content and functionality chunks. The content portfolio is used for collecting information requirements. It is based on information needs and demands, and links the storyboard to the lower-level conceptual model of the WIS consisting of a collection of media types. The utilisation portfolio is used

for collecting functionality requirements. It describes intentions of users, specific needs and their context.

In addition to this follow-on part on storyboarding pragmatics we are also working on the pragmatism of storyboarding, storyboard refinements, and quality evaluation. All this together plus the ongoing research on logical grounds of storyboarding and their exploitation for reasoning and verification will complete our research on high-level WIS design within the codesign framework.

Bibliography

1. AKAISHI, M., SPYRATOS, N., AND TANAKA, Y. A component-based application framework for context-driven information access. In *Information Modelling and Knowledge Bases XIII*, H. Kangassalo et al., Eds. IOS Press, 2002, pp. 254–265.
2. ALTUS, M. User modelling for conceptual database design based on an extended entity relationship model: A preliminary study relationship model. In *Proc. ADBIS* (1996), pp. 46–51.
3. AMBLER, S. W. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. John Wiley & Sons, 2002.
4. BENYON, D., GREEN, T., AND BENTAL, D. *Conceptual modeling for user interface development*. Springer, London, 1998.
5. BJORNER, D. *Software Engineering 3: Domains, requirements, and software design*. Springer, Berlin, 2006.
6. BOUQUET, P., SERAFINI, L., BREZILLON, P., BENERECETTI, M., AND CASTELLANI, F., Eds. *Modeling and Using Context – Context'99* (1999), vol. 1688 of *LNAI*, Springer-Verlag.
7. CARROLL, J. M., Ed. *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press, Cambridge, England, 1991.
8. CARROLL, J. M. Participatory design of community information systems - the designer as bard. In *COOP* (2004), pp. 1–6.
9. CERI, S., FRATERNALI, P., BONGIO, A., BRAMBILLA, M., COMAI, S., AND MATERA, M. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, San Francisco, 2003.
10. CONALLEN, J. *Building Web Applications with UML*. Addison-Wesley, Boston, 2003.
11. CONSTANTINE, L., AND LOCKWOOD, L. *Software for use: a practical guide to the models and methods of usage-centered design*. Addison-Wesley, Reading, Massachusetts, 1999.
12. COURAGE, C., AND BAXTER, K. *Understanding your users: a practical guide to user requriements - methods, tools & techniques*. Morgan Kaufman, Boston, 2005.
13. DE TROYER, O., AND LEUNE, C. WSDM: A user-centered design method for web sites. In *Computer Networks and ISDN Systems – Proceedings of the 7th International WWW Conference*. Elsevier, 1998, pp. 85–94.
14. GIORGINI, P., MYLOPOULOS, J., NICCHIARELLI, E., AND SEBASTIANI, R. Reasoning with goal models. In *ER* (2002), pp. 167–181.
15. G.P. PFAFF, P. H. User interface management systems. Springer, Heidelberg, 1985.
16. GÜELL, N., SCHWABE, D., AND VILAIN, P. Modeling interactions and navigation in web applications. In *Conceptual Modeling for E-Business and the Web*, S. W. Liddle, H. C. Mayr, and B. Thalheim, Eds., vol. 1921 of *LNCS*. Springer-Verlag, 2000, pp. 115–127.
17. HAREL, D., AND MARELLY, R. *Come, Let's play: Scenario-based programming using LSCs and the play-engine*. Springer, Berlin, 2003.
18. HECKMANN, D., AND KRÜGER, A. A user modeling markup language (userml) for ubiquitous computing. In *User Modeling* (2003), pp. 393–397.

19. HECKMANN, D., SCHWARTZ, T., BRANDHERM, B., SCHMITZ, M., AND VON WILAMOWITZ-MÖLLENDORFF, M. Gumo - the general user model ontology. In *User Modeling* (2005), vol. 3538 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 428–432.
20. HEINRICH, L. J. *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur*. Oldenbourg Verlag, München, 1996.
21. HOUBEN, G.-J., BARNA, P., FRASINCAR, F., AND VDOVJAK, R. HERA: Development of semantic web information systems. In *Third International Conference on Web Engineering – ICWE 2003* (2003), vol. 2722 of *LNCS*, Springer-Verlag, pp. 529–538.
22. KASCHEK, R., SCHEWE, K.-D., THALHEIM, B., AND ZHANG, L. Integrating context in conceptual modelling for web information systems. In *Web Services, E-Business, and the Semantic Web*, C. Bussler, D. Fensel, M. E. Orlowska, and J. Yang, Eds., vol. 3095 of *LNCS*. Springer-Verlag, 2004, pp. 77–88.
23. KENSING, F., AND BLOMBERG, J. Participatory design: Issues and concerns. *Computer Supported Cooperative Work 7*, 3/4 (1998), 167–185.
24. KOBSA, A. User modeling and user-adapted interaction. *User Modeling and User-Adapted Interaction 14*, 5 (2004), 469–475.
25. KOBSA, A. User modeling and user-adapted interaction. *User Modeling and User-Adapted Interaction 15*, 1-2 (2005), 185–190.
26. LOWE, D., HENDERSON-SELLERS, B., AND GU, A. Web extensions to UML: Using the MVC triad. In *Conceptual Modeling – ER 2002*, S. Spaccapietra, S. T. March, and Y. Kambayashi, Eds., vol. 2503 of *LNCS*. Springer-Verlag, 2002, pp. 105–119.
27. MAGNINI, B., AND STRAPPARAVA, C. User modelling for news web sites with word sense based techniques. *User Modeling and User-Adapted Interaction 14*, 2-3 (2004), 239–257.
28. MORITZ, T., SCHEWE, K.-D., AND THALHEIM, B. Strategic modelling of web information systems. *International Journal on Web Information Systems 1*, 4 (2005), 77–94.
29. MYLOPOULOS, J., FUXMAN, A., AND GIORGINI, P. From entities and relationships to social actors and dependencies. In *Conceptual Modeling - ER 2000* (Berlin, 2000), Springer-Verlag, pp. 27–36.
30. MYLOPOULOS, J., AND MOTSCHNIG-PITRIK, R. Partioning information bases with contexts. In *Proc. CoopIS '95*. 1995, pp. 44–55.
31. O'NEILL, E., AND JOHNSON, P. Participatory task modelling: users and developers modelling users' tasks and domains. In *TAMODIA* (2004), pp. 67–74.
32. PAECH, B. *Aufgabenorientierte Softwareentwicklung*. Springer, Berlin, 2000.
33. RAZMERITA, L., ANGEHRN, A. A., AND MAEDCHE, A. Ontology-based user modeling for knowledge management systems. In *User Modeling* (2003), pp. 213–217.
34. ROBERTSON, J., AND ROBERTSON, S. *Mastering the Requirements Process*. Addison-Wesley, 1999.
35. ROBERTSON, J., AND ROBERTSON, S. *Requirements-Led Project Process*. Addison-Wesley, 2006.
36. ROSENFELD, L., AND MORVILLE, P. *Information Architecture*. O'Reilly, Cambridge, 1998.
37. ROSSI, G., SCHWABE, D., AND LYARDET, F. Web application models are more than conceptual models. In *Advances in Conceptual Modeling*, P. Chen et al., Eds., vol. 1727 of *LNCS*. Springer-Verlag, Berlin, 1999, pp. 239–252.
38. SCHEWE, B. *Kooperative Softwareentwicklung*. Deutscher UniversitätsVerlag, Wiesbaden, 1996.
39. SCHEWE, K.-D., AND THALHEIM, B. Life cases: An approach to address pragmatics in the design of web information systems. submitted for publication.

40. SCHEWE, K.-D., AND THALHEIM, B. The co-design approach to web information systems development. *International Journal on Web Information Systems 1*, 1 (2005), 5–14.
41. SCHEWE, K.-D., AND THALHEIM, B. Conceptual modelling of web information systems. *Data and Knowledge Engineering 54*, 2 (2005), 147–188.
42. SCHEWE, K.-D., AND THALHEIM, B. User models: A contribution to pragmatics of web information systems design. In *Web Information Systems – Proceedings WISE 2006*, K. Aberer, Z. Peng, and E. Rundensteiner, Eds., vol. 4255 of *LNCS*. Springer-Verlag, 2006, pp. 512–523.
43. SCHEWE, K.-D., AND THALHEIM, B. Personalisation of web information systems - a term rewriting approach. *Data and Knowledge Engineering* (2007). to appear.
44. SCHEWE, K.-D., THALHEIM, B., BINEMANN-ZDANOWICZ, A., KASCHEK, R., KUSS, T., AND TSCHIEDEL, B. A conceptual view of electronic learning systems. *Education and Information Technologies 10*, 1-2 (2005), 83–110.
45. SCHWABE, D., AND ROSSI, G. An object oriented approach to web-based application design. *TAPOS 4*, 4 (1998), 207–225.
46. SOMMERVILLE, I. *Software Enginering*, seventh ed. Addison Wesley, San Francisco, 2004.
47. THALHEIM, B., AND DÜSTERHÖFT, A. The use of metaphorical structures for internet sites. *Data & Knowledge Engineering 35* (2000), 161–180.
48. THEODORAKIS, M., ANALYTI, A., CONSTANTOPOULOS, P., AND SPYRATOS, N. Context in information bases. In *Proc. CoopIS '98*. 1998, pp. 260–270.
49. THEODORAKIS, M., ANALYTI, A., CONSTANTOPOULOS, P., AND SPYRATOS, N. Contextualization as an abstraction mechanism for conceptual modelling. In *Conceptual Modeling – Proc. ER'99* (1999), vol. 1728 of *LNCS*, Springer-Verlag, pp. 475–489.
50. VALE, E. *The technique of screen and television writing*. Simon and Schuster, New York, 1982.
51. Webster's ninth new collegiate dictionary, 1991.