Christian-Albrechts-Universität zu Kiel
Technische Fakultät
Institut für Informatik

Dissertation zur Erlangung des Grades einer Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

Reference Model and Architecture for the Post-Platform Economy

vorgelegt von

Mirjana Radonjić-Simić

Juni 2022

1. Betreuer:	Prof. Dr. Andreas Speck
2. Betreuer:	Prof. Dr. Stefan Fischer

Datum der mündlichen Prüfung: 30. November 2022

Acknowledgements

I want to express my gratitude to many people without whose help and support this work would not have been possible. First of all, I would like to thank my supervisor, Prof. Dr. Andreas Speck, for his extraordinary support, continuous motivation, and valuable discussions. I would also like to thank my co-supervisor, Prof. Dr. Stefan Fischer, for supporting me and taking the time to comment on my thesis.

My special thanks go to Prof. Dr.-Ing. Dennis Pfisterer for numerous valuable discussions. His detailed and timely feedback helped me structure and refine my ideas and complete this thesis. I would also like to thank Prof. Dr.-Ing. Jörg Baumgart, who encouraged me to start and carry out this work, and other colleagues from the Duale Hochschule Baden-Württemberg and the Christian-Albrechts-University zu Kiel for many fruitful discussions and valuable comments. Especially to Dr.-Ing. Michael Hitz, Prof. Dr.-Ing. Clemens Martin, Prof. Dr. Frank Wolff, Prof. Dr. Julian Reichwald, Petko Rutešić, Melanie Windrich, Dr.-Ing. Sven Niemand and Christian Mahrt.

Finally, my great gratitude goes to those with whom I share friendship and deep love - my friends for their support over many years and my dear husband Saša, who has always encouraged me in my work and offered patience and understanding in stressful times. And to my beloved son Filip, who helped me to recognize the essential things in life. I am incredibly grateful to my parents for their love and enduring support.

Kurzfassung

Die rasante Entwicklung des Internets und der damit verwandten Technologien haben neuartige Geschäftsmodelle ermöglicht. Es sind Plattformunternehmen (Plattformen) entstanden, die eine breite Palette menschlicher Aktivitäten über das Internet ermöglichen und somit die Art und Weise, wie wir arbeiten, uns vernetzen und geschäftliche Werte schaffen, radikal verändern. Plattformen wie Amazon, Alibaba und Uber bilden die Basis der gegenwärtigen Plattformökonomie, die unterschiedliche soziale und wirtschaftliche Aktivitäten ermöglichen, die wiederum durch die den Plattformen zugrundeliegenden Geschäftsmodelle geprägt sind. Aufgrund ihres enormen Erfolgs konzentrieren sich heute mehr Angebote als je zuvor auf Plattformen. Dies untermauert die Machtposition der Plattformen und versetzt sie in eine Monopolstellung, in der sie Regeln diktieren, den Zugang kontrollieren und so zu einer faktischen Zentralisierung von zuvor dezentralisierten Angeboten im Internet führen können. Eine weitere Herausforderung im Zusammenhang mit der modernen Plattformökonomie besteht darin, dass Plattformen zwar sehr gut einzelne Produkte und Dienstleistungen abwickeln, aber keine komplexen Produkte unterstützen (d. h. personalisierte Kombinationen einzelner Produkte oder Dienstleistungen, die bestimmte vom Verbraucher definierte Kriterien erfüllen sollen). Folglich müssen Verbraucher, die nach komplexen Produkten suchen, wissen, wo und wie sie die optimale Produkt-/Dienstleistungskombination finden, wodurch die Koordinations- und Transaktionskosten für solche Produkte steigen.

Um diese Herausforderungen zu adressieren, wird in dieser Arbeit eine Post-Plattform-Ökonomie konzipiert. In dieser Ökonomie verlagert sich die Macht weg von den Plattformen hin zu den Verbrauchern und Anbietern als den Hauptakteuren des wirtschaftlichen Austauschs im Internet. Eine solche Ökonomie erkennt die Bedeutung der Verbraucher und ihrer persönlichen Bedürfnisse an. Sie ermöglicht es allen Agierenden, die mit dem Internet verbunden ist, dazu beizutragen und solche komplexen Bedürfnisse zu erfüllen, entweder als Verbraucher oder als Anbieter oder beides gleichzeitig. Folglich bezieht sich die Post- Plattformökonomie auf wirtschaftliche Aktivitäten, die durch selbstorganisierte und streng dezentralisierte Online-Strukturen, die sogenannten „Distributed Market Spaces", ermöglicht werden. Das Hauptziel der Distributed Market Spaces besteht somit darin, den negativen Auswirkungen der wachsenden Macht der etablierten Plattformen entgegenzuwirken und die Koordinations- und Transaktionskosten für komplexe Produkte zu senken, während gleichzeitig die Vorteile moderner Plattformen erhalten bleiben.

Das Ziel dieser Arbeit ist es, ein Referenzmodell und eine begleitende Softwaresystemarchitektur zu entwerfen, die zusammen als Anwendungsframework für die Analyse, das Design und die Implementierung von Distributed Market Spaces dienen können.

Der Nutzen eines solchen Anwendungsframeworks ist zweifach: Zum einen liefert es Erkenntnisse, die für das Verständnis verschiedener Aspekte und Elemente selbstorganisierter und streng dezentraler Online-Strukturen unerlässlich sind. Zum anderen dient es als Blaupause für die Gestaltung und Implementierung einer Instanz eines Distributed Market Space in einem bestimmten Anwendungskontext. Damit werden Verbraucher und Anbieter unterstützt, selbst Market Spaces aufzubauen, in denen sie direkt und zuverlässig komplexe Produktszenarien abwickeln können.

Diese Arbeit thematisiert die gesetzte Zielsetzung in folgender Weise: Erstens führt sie Distributed Market Spaces als neues Konzept für die Post-Plattformökonomie ein. Dabei werden die Charakteristika und allgemeinen Anforderungen sowie der Geltungsbereich der Distributed Market Spaces definiert und demzufolge als Domäne für die Referenzmodellierung identifiziert. Zweitens wird ein Referenzmodell für Distributed Market Spaces entwickelt, welches das Konstrukt des Distributed Market Space (Referenz-DMS) aus drei verschiedenen Perspektiven beschreibt: Phases, Views und Stages. Das vorgeschlagene Referenzmodell definiert wie eine Referenz-DMS die Markttransaktionen für komplexe Produkte unterstützt (Phases), wie sie auf strategischer sowie operativer Ebene organisiert ist (Views) und wie sich ihre Instanzen im Lebenszyklus entfalten können (Stages). Drittens wird eine Architektur für Distributed Market Spaces entwickelt und prototypisch implementiert. Die vorgeschlagene Architektur ist eine streng dezentrale und hoch skalierbare Softwaresystemarchitektur, die für die konkrete Implementierung eines Referenz-DMS dienen kann. Sie definiert die Struktur eines Informationssystems, das erforderlich ist, um eine Referenz-DMS auf der operativen Ebene zu implementieren. Schließlich wird die Anwendbarkeit des vorgeschlagenen Referenzmodells und der dazugehörigen Architektur demonstriert und in zwei verschiedenen Anwendungsszenarien evaluiert. Die beiden Anwendungen der Smart City und des COVID-19 Pandemiemanagement wurden gewählt, um die vielfältigen Einsatzmöglichkeiten des vorgeschlagenen Referenzmodells für Distributed Market Spaces zu veranschaulichen.

Abstract

The rapid development of the Internet and related technologies have opened up new opportunities for creating novel business models. Using technology, companies are creating platform companies that enable a wide range of human activities over the Internet, radically changing the way we work, connect, and create value in business. Platforms such as Amazon, Alibaba, and Uber form the basis of the platform economy – a subset of social and economic activities made possible by platform business models. Because of their tremendous success, more offers than ever are now centered around platforms. This underpins the platforms' positional power and puts them in a monopoly position. They can dictate rules, control access and thus lead to a de facto centralization of previously decentralized offers on the Internet. Another issue related to the modern platform economy is that platforms work well for single products and services, but do not support complex products (i.e., personalized combinations of individual products or services that have to meet specific consumer-defined criteria). Consequently, consumers searching for complex products need to know where and how to find the optimal product/service combination, consequently increasing transaction costs for such products.

To address these issues, this work envisions a post-platform economy. This economy shifts power from platforms to consumers and providers as the primary drivers of economic exchanges on the Internet. Such an economy recognizes the importance of consumers and their personalized requirements. It enables anyone and anything connected to the Internet to contribute and meet such complex needs, either as a consumer or a provider, or both simultaneously. Consequently, the post-platform economy refers to economic activities enabled by self-organized and strictly decentralized online structures, the so-called "distributed market spaces". The main goal of distributed marketplaces is to counteract the adverse effects of growing platform power and lower transaction costs for complex products while preserving the advantages and opportunities of modern platforms.

The primary goal of this thesis is to propose a reference model and an accompanying software system architecture, which together can serve as a guiding framework for the analysis, design, and implementation of distributed market spaces. The benefit of such a framework is considered two-fold: On the one hand, it provides insights essential for understanding various aspects and elements of self-organized and strictly decentralized online structures to facilitate the emergence of the post-platform economy. On the other hand, it serves as a blueprint for designing and implementing a distributed marketplace instance for a specific application context. It thus allows consumers and providers to set up and expand market spaces themselves, in which they can engage directly and reliably with complex product scenarios.

This work makes the following contributions: First, it introduces distributed market spaces as a new concept for the post-platform economy. It identifies their primary characteristics and overall objectives and defines the scope of distributed market spaces as a domain of interest for reference modeling. Secondly, a reference model for distributed market spaces is developed, which describes the construct of distributed market spaces (reference DMS) from three different perspectives: phases, views, and stages. As a multi-dimensional and multi-view reference model, it defines how a reference DMS facilitates market transactions for complex products (phases), how it works on a strategic and operational level (views), and how its instances can unfold during the life cycle (stages). Third, it designs and develops an architecture for distributed marketplaces as a strictly decentralized and highly scalable software system architecture for a concrete implementation of a reference DMS. It represents a blueprint of an information system required to implement a reference DMS at the operational level and thus to realize foundational, governance, and specialized services as defined by the service view of the reference model. Finally, it demonstrates the applicability of the proposed reference model and accompanying architecture and evaluates them in two different application scenarios. The two applications of Smart City and COVID-19 pandemic management are chosen to illustrate the wide variety of potential applications of the proposed reference model for distributed market spaces.

Contents

Indexes

List of Figures

List of Tables

Listings

Abbreviations

A_DG	Architecture Design Goal
CP	Complex Product
CPC	Complex Product Contract
CPP	Complex Product Proposal
CPR	Complex Product Request
DHT	Distributed Hash Table
DMS	Distributed Market Spaces
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
O	Objective
PC	Pending Contract
R	Requirement
RDF	Resource Description Framework
RfO	Request for Offer
RM_DG	Reference Model Design Goal
SPARQL	SPARQL Protocol and RDF Query Language

WebRTC	Web Real-Time Communication

1 | Introduction

The fast development of the Internet and related technologies bring unprecedented new opportunities. By leveraging technology, companies are creating structures that enable a wide range of human activities over the Internet. These online structures open up the way for radical changes in the way we work, socialize, and create value in the economy [PVC16]. Such structures refer to online platforms and form the basis for the "platform economy" – a subset of social and economic activities enabled by online platforms [KZ16].

Amazon, Alibaba, eBay, Airbnb, and Uber represent some prominent and tremendously successful businesses built as platforms. Even though these platforms pursue different strategies, have different functions, and operate in various domains, each merely connects other user groups and enables them to exchange value. For example, Amazon, Alibaba, and eBay connect consumers looking for products and producers offering certain products, while Airbnb and Uber connect consumers looking for a particular service like accommodation or a taxi.

However, the term “platform” has evolved over the years and is currently used to describe different things in many contexts. Platforms generally connect people and organizations for a common purpose or share of a common resource [PVC16], [MJ16], [ES16]. This thesis follows an extended definition proposed by Cusumo et al. [CGY19], defining a platform ”*as a technology-based organization model that brings together individuals and organizations so they can interact and innovate in ways not otherwise possible, with the potential for nonlinear increases in utility and value*”.

Depending on their primary function, existing platform models can be divided into two basic types: innovation and transaction-oriented platforms [PVC16], [CGY19]. Innovation platforms usually consist of technological building blocks shared and used to create new complementary products or services (e.g., Apple iOS or Amazon AWS). By contrast, transaction platforms are mainly large intermediaries that enable people and organizations to exchange information (e.g., Facebook, Twitter), or buy, sell, or access various goods and services (e.g., Amazon and Uber).

As models for market exchange, platforms are not new. They are going all the way back to early marketplaces, from bazaars up to modern shopping malls, and auction houses. However, the main difference is that platforms as online structures can facilitate market exchange at Internet scale without any time or location constraints, and thus on an unprecedented scale [RR17], [PVC16].

Platforms are based on technology, but they are more than a piece of technology or a tool. Instead, platforms are business models that facilitate the exchange of value between multiple user groups; usually, consumers and producers [MJ16]. To make these exchanges happen, they create and harness large, scalable networks of users and resources accessed on-demand. That is the reason why platforms are also characterized as two or multisided markets [RT03] or matchmakers [ES16]. The term matchmaker indicates the platform's overarching purpose of bringing together different user groups of consumers and providers and supporting them to interact and exchange value [PVC16]. Thus, the value exchanged can be products, services, knowledge, information, and currency, while enabling value creation for all participants. Figure 1.1 illustrates the anatomy of a platform business model.

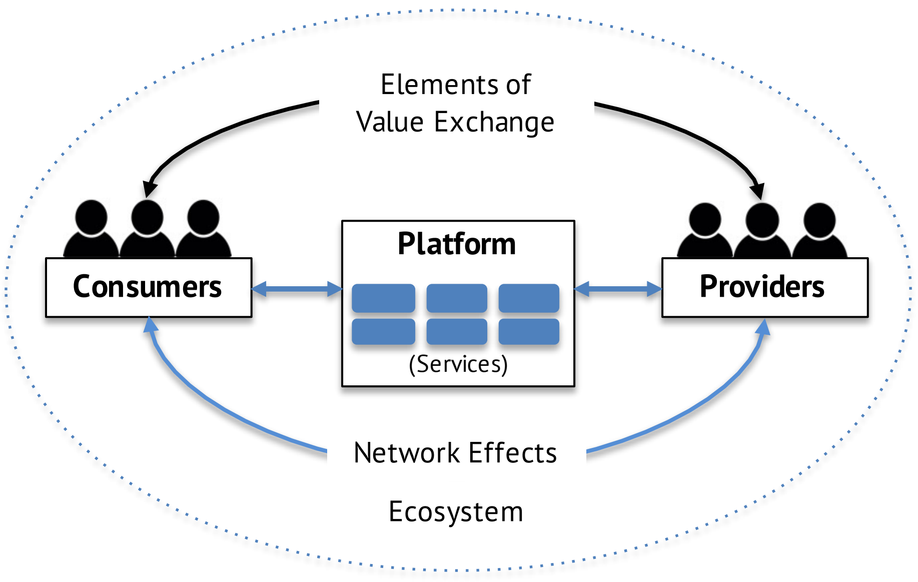


Figure 1.1: Anatomy of a platform business model (adopted from [PVC16])

Creating value by providing an open, participative functional infrastructure is at the heart of platform business models. As shown in the middle of Figure 3.2, the provided functional infrastructure represents "the platform" where consumers and providers can plugin in order to interact and transact with each other. The platform is represented by a set of services, which realize the functions that a platform needs to offer on the operational level. As to [ES16], [PVC16], [RR17], such a set of services realize the core platform's functions related to creating, connecting, consuming, compensating, and optimizing. In other words, the core functions of a platform are considered the following:

- attracting the audience (building a liquid marketplace by attracting a critical mass of consumers and providers)
- matchmaking (connecting right consumers with right providers to facilitate transactions and interactions)
- providing tools and services, and
- establishing rules and standards to govern the network of participants and, thus, build trust and maintain quality

For the involved consumers and providers, the platform's core functions significantly reduce transaction costs, whereby transaction costs refer to any costs incurred in participating in interactions and making a market exchange [Coa37].

Furthermore, by providing tools and establishing rules and standards relevant to building interactive networks, platforms empower the creation of ecosystems, also called "value ecosystems" (see Figure 1.1). These ecosystems are the source of value capture for the platform business models. Platforms benefit from their ecosystems by taking advantage of the idea of network effects [PVC16]. In the context of a platform business model, network effects indicate that a user's behavior has a direct impact on the value that other users receive from the same network [KS94]. Network effects refer to the effect that the increased number of users has on the platform's value to any given user [EPV06], [PV05]. For example, video game developers (i.e., providers) will create games only for platforms with a significant number of players (i.e., consumers), as they need a large enough consumer base to recapture their development costs. Players, in turn, prefer platforms with a broader collection of games.

Consequently, the user in a platform's network is also a co-creator of value to the network. Such positive network effects are the primary source of value creation and competitive advantage in the platform businesses. Platforms leverage the positive network effects in their ecosystems through charging fees, advertising, and monetizing data collected across the network [ES16]. However, network effects can also be harmful, as the growth of a network can produce adverse effects, which might push away users from the platform. The most successful platform businesses, however, minimize adverse network effects by the process of effective curation [PV05] – a governance mechanism by which a platform controls the access of the users of the platform, the activities they participate in, and how they connect.

Businesses built as platforms have grown at a tremendous pace and are thriving all over the world [CGY19]. According to the Forbes 2000 list [For] Amazon, for example, offers more than 250 million products around its marketplace, eBay manages more than 300 thousand transactions daily. Alibaba, as a mega-platform, encompassing Alipay, AliExpress, Taobao, and Tmall dominates the Asia market with more than 80 percent of market share [PVC16].

This tremendous success of platform businesses has broader implications for the organization of global markets and value creation in the economy as a whole. The following section will discuss some of these implications and related challenges, and reveal how this thesis attempts to approach them.

1.1 Motivation

As to the considerations above, many platforms leverage technology and the economies of network effects in a way as to grow large value ecosystems and gain unprecedented power [RR17], [PVC16]. This power derives from their intermediary position and ability to leverage a massive amount of data collected from the interactions and transactions within their ecosystems. Even though ecosystems include networks of users, which are distributed and not under the direct control of the platform, those networks are planned and orchestrated in a centralized manner [MJ16]. As a result, the information processing and aggregation necessary for the functioning of these networks are centralized and exclusively controlled by the underlying platforms. This underpins the positional power of platforms, putting them in a position of a monopoly [MJ16], where they can dictate the rules, control access, and thus lead to a de-facto centralization of previously decentralized offerings on the Internet. More and more initiatives, therefore, call for a re-decentralization of the World Wide Web (e.g., SOLID Project by Berners-Lee [Ber17]), and for the re-decentralization of the Internet as a global market space (e.g., OpenBazaar [Baz16]). In 2020, the European Commission proposed the Digital Markets Act (DMA)[Comb] as part of the European digital strategy to regulate the positional power of platforms. DMA is a part of ”A Europe fit for the digital age” – one of the top six priorities for 2019-2024. It argues that ... ”Europe must now strengthen its digital sovereignty and set standards, rather than following those of others – with a clear focus on data, technology, and infrastructure.”[Coma]. Thus, standards from others refer to the standards of big platforms initially located in the USA and Asia, as previously mentioned. Another issue related to the modern platform economy is that platforms work well for individual products and services, but fail to support consumers looking for personalized combinations of individual products and services to satisfy a particular need. Such personalized bundles of different products and services are considered complex products.

Definition 1.1: *Complex product* refers to an arbitrary combination of individual products and services that need to fulfill a particular consumer-defined context in order to satisfy a personalized need.

Consider, for example, a couple who wants to spend a pleasant evening with friends at the theatre. As a consumer, this couple requires a combination of services that includes the following: tickets for the theatre, reservation of a table at an Italian restaurant, finding parking spot close to both locations, and engaging a babysitter to watch after their children. These requirements span four different service domains (i.e., ticketing, gastronomy, parking, and babysitting), and considers contextual information regarding the schedule, location, and ratings of a particular service. For our couple, those could be engaging a babysitter from a well-rated and certified babysitting service for the whole evening, reserving a table with certain restrictions (e.g., availability of a seaside terrace), and parking spot with less than 200m walking distance).

For each of these domains, several platforms exist. However, they focus solely on products and services from the supported domains (e.g., Eventim for domain ticketing and MyTable for gastronomy), which is why platforms are often called ”’verticals”. Exceptions are platforms that offer pre-defined combinations of products and services traditionally brought together (e.g., Opodo supporting bundled transactions of a flight, hotel, rental car, and travel assurance). Figure 1.2 illustrates the vertical orientation of contemporary platforms spanning over domains ($Domain_1$, ..., $Domain_n$). For our couple, it implies that it needs to combine different verticals in order to get the required service combination, put it all together, and all of this in the context of schedule

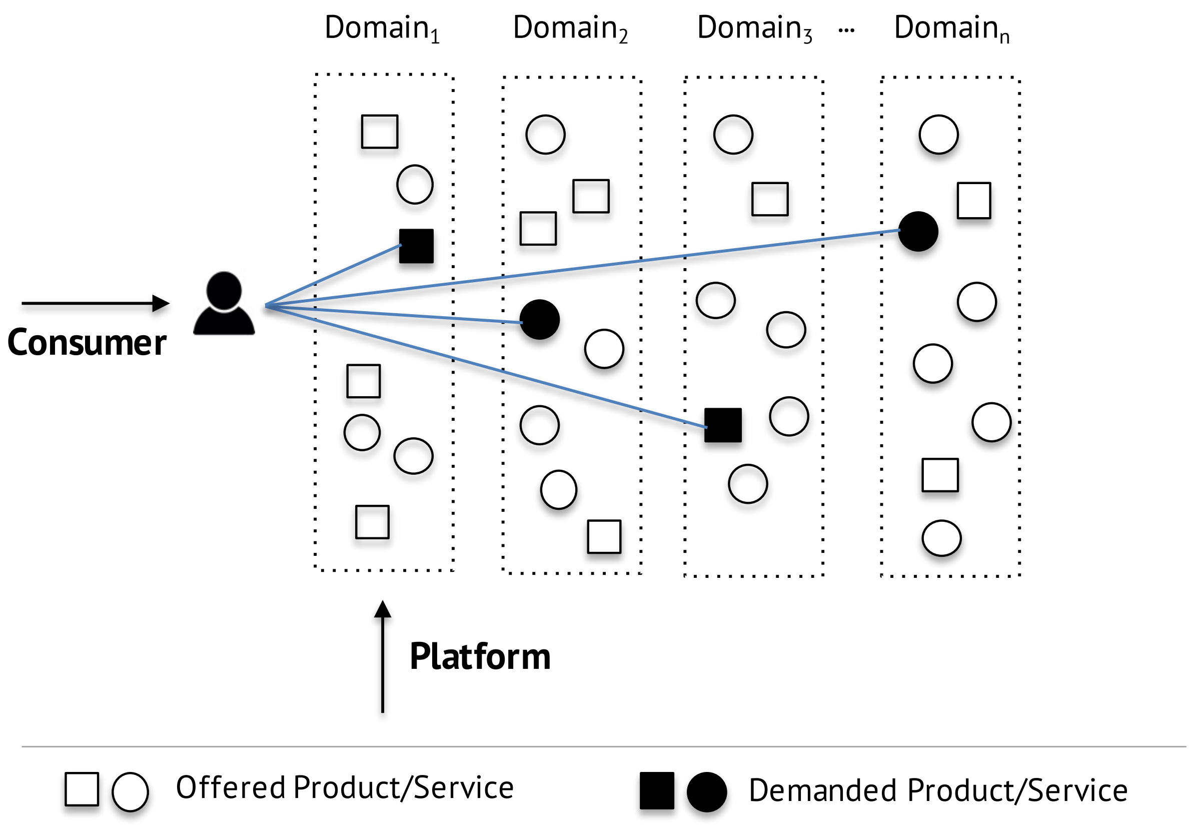


Figure 1.2: Complex products in platform economy

and location. This requires a high level of personal involvement and manual activities, which contradicts the primary purpose of platforms as matchmakers [ES16]–connecting the right consumers with the right providers and, by doing so, significantly reducing transaction costs. Furthermore, this complexity of finding personalized product/service combinations overstrains consumers and leads to decisions according to the principle of adverse selection [Ant16]. In such cases, consumers tend to choose ”good enough” instead of ”optimal”, increasing transaction costs for complex products.

To address the identified issues, this work envisions a **Post-Platform Economy**. An economy that shifts the power from platforms to consumers and providers as the primary drivers of market exchange on the Internet by:

- recognizing the importance of consumers and their personalized requirements,
- enabling everyone and everything connected to the Internet to contribute to satisfying such personalized requirements,
- being a consumer or a provider, or both at the same time.

In addition to that, in **Post-Platform Economy**, consumers and providers are considered equal in their rights and responsibilities as they can engage in complex product scenarios directly without any intermediaries and related barriers and constraints. As market participants, they are constitutive parts, as they intend to participate in market exchange and provide for the underlying market mechanisms. For example, they share resources and provide services to build supportive infrastructures considered necessary for an economy that does not rely on centrally orchestrated structures to enable market exchange.

Definition 1.2: *Post-Platform Economy* refers to economic activities facilitated by self-organized and strictly decentralized online structures termed as *Distributed Market Spaces*.

The main idea of *Distributed Market Spaces* is, therefore:

- to counter the adverse effects of growing platform-power, and
- to lower transaction costs for complex products,

while maintaining the benefits and enabling the nature of centrally orchestrated platform models.

The primary goal of this thesis is to propose:

- *a reference model and*
- *an accompanying software-system architecture*

which together can serve as a guiding framework for the analysis, design, and implementation of distributed market spaces.

The benefit of such a *framework for distributed market spaces* is considered twofold: Firstly, it can provide insights essential for understanding different aspects, entities, and elements of distributed market spaces as self-organized and strictly decentralized online structures to facilitate the emergence of the post-platform economy. Secondly, it serves as a blueprint on how to conceptualize and implement distributed market spaces for a specific application context in the post-platform economy. Hence, it intends to enable market participants to establish and enhance value ecosystems on their own, where they can engage in complex product scenarios directly and reliably.

1.2 Approach

The modeling process employed to develop the reference model for distributed market spaces follows the model by Schütte [Sch13b] (cf. Section 2.1.2). It is utilized as a starting point, but since this thesis also aims to develop an accompanying architecture, the Schütte's model is extended to integrate the design and development process of the architecture for distributed market spaces.

As illustrated in Figure 1.3, the construction process for the reference model, design and development of the architecture for distributed market spaces includes five phases:

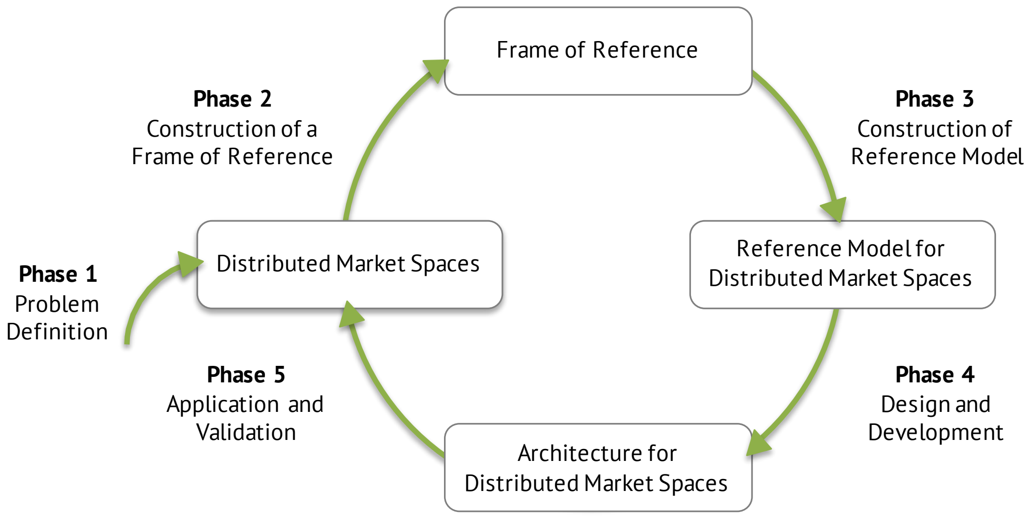

Frame of Reference
Phase 2
Construction of a Frame of Reference
Phase 3
Construction of Reference Model
Phase 1
Problem Definition
Distributed Market Spaces
Reference Model for Distributed Market Spaces
Phase 5
Application and Validation
Phase 4
Design and Development
Architecture for Distributed Market Spaces

Figure 1.3: Modeling process for reference model and architecture for distributed market spaces (based on [Sch13b])

- **Phase 1** Problem Definition – at the beginning of the construction process, the domain of the distributed market spaces is determined. This includes a definition of the scope, a brief description of the characteristics, and a description of the intended use and the overall objectives of distributed market spaces.
- **Phase 2** Construction of a Frame of Reference – the starting point for the modeling activities is in the development of a frame of reference. This phase has two functions: Supporting the systematic identification of the required model modules and elements. On the other, the processing of its structure contributes to the completeness of the intended reference model.
- **Phase 3** Construction of Reference Model – in this phase, the reference model for distributed market spaces is constructed based on identified building blocks and elements by considering construction principles. It is gradually refined in terms of inter-model consistency.
- **Phase 4** Design and Development – after completing the construction of the reference model, a software system architecture is designed and developed to support a concrete implementation of the constructed reference model.
- **Phase 5** Application and Validation – finally, the proposed reference model and architecture for distributed market spaces are applied in the context of two application scenarios. The main focus is on validating whether the constructed reference model meets the requirements and fulfills the intended purpose for which it was designed.

1.3 Contributions and Structure of this Thesis

This thesis makes the following contributions:

1. It introduces *distributed market spaces* as a new concept for the post-platform economy. It identifies their primary characteristics and overall objectives and defines the scope of distributed market spaces as a domain of interest for reference modeling [RP15], [PRR16], [RWP17].
2. It develops a *reference model for distributed market spaces* for the analyse and design of self-organized structures that lower transaction costs for complex products and facilitate market exchange in a decentralized manner [RWP17], [RP19a], [RP19b], [RP19c].
3. It designs, develops and evaluates an *architecture for distributed market spaces* as a strictly decentralized and highly scalable software-system architecture for a concrete implementation of a distributed market space [PRR16], [Hit+16], [RP19c], [RPR20], [RRP21], [RP19b].
4. It demonstrates the applicability and feasibility of the proposed reference model for distributed market spaces and the accompanying architecture evaluating them in the context of different application scenarios [RP19b], [Rad+21].

Figure 1.4 visualizes these contributions and illustrates which contributions build on each other and how they relate to the structure of this thesis.

The remainder of this thesis is structured as follows:

- Chapter 2 gives an overview of the fundamentals of this thesis. It first overviews the concept of reference models and reference modeling and introduces different approaches and design principles. Then, it overviews the peer-to-peer systems, their architectures, and primary components that use the decentralized organization principle to share information and resources.
- Chapter 3 introduces distributed market spaces as a new concept for the post-platform economy, formally defining the terms and terminologies used in this thesis. It presents the main characteristics of distributed market spaces derived from the two main driving forces decentralization and complex products. It analyzes complex product scenarios using the BOAT [Gre15] as a method for scenario analysis. It identifies the overall objectives of distributed market spaces.
- Chapter 4 introduces the constructed reference model for distributed market spaces. It presents its structure and the constitutive elements grouped in three dimensions, four views, and five phases. It provides a detailed description of these elements and the underlying models. It defines how a distributed market space as an organizing model works on the strategic and operational level, enables market transactions for complex products, and how its instances might unfold during different life stages.

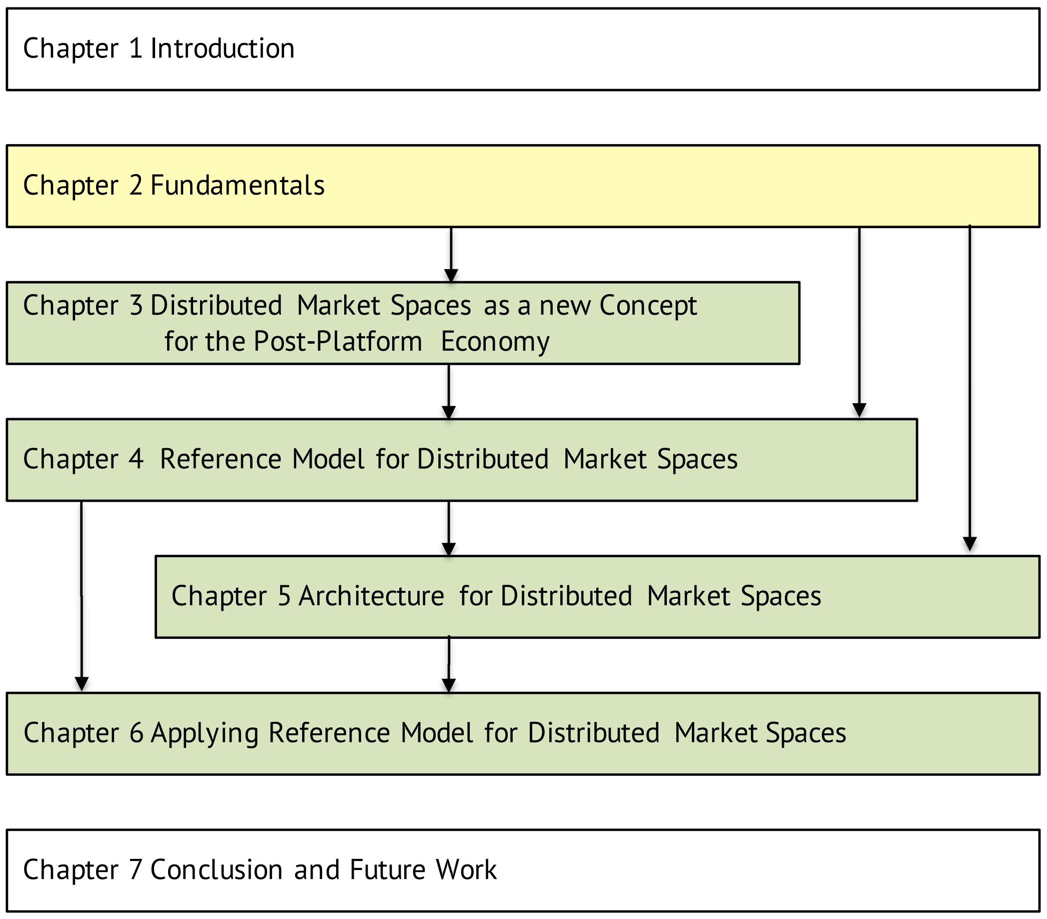

Chapter 1 Introduction
Chapter 2 Fundamentals
Chapter 3 Distributed Market Spaces as a new Concept for the Post-Platform Economy
Chapter 4 Reference Model for Distributed Market Spaces
Chapter 5 Architecture for Distributed Market Spaces
Chapter 6 Applying Reference Model for Distributed Market Spaces
Chapter 7 Conclusion and Future Work

Figure 1.4: Visualization of contributions of this thesis, their relations to each other, and to the structure of the thesis

- Chapter 5 designs, develops and evaluates the architecture for distributed market spaces as a possible implementation of the infrastructure view of the proposed reference model. It outlines the primary building blocks, followed by the description of the functional structure, its core components, and their interaction, followed by the specification of primary information structures. It presents a proof-of-concept implementation and evaluates it applying the ATAM [CKK+03] scenario-based evaluation method.
- Chapter 6 applies the reference model for distributed market spaces and the accompanying architecture. Using two application scenarios, it demonstrates how decentralized and self-organized online structures could be analyzed, designed, and implemented using the reference model for distributed market spaces as guidance. It discusses the reference model's ability to fulfill the intended purpose and meet the design goals for which it is constructed.
- Chapter 7 summarizes and discusses the contributions of this thesis and gives an outlook on future work.

2 | Fundamentals

This chapter provides the fundamentals necessary for understanding this thesis and its contributions. Figure 2.1 visualizes the structure of this chapter.

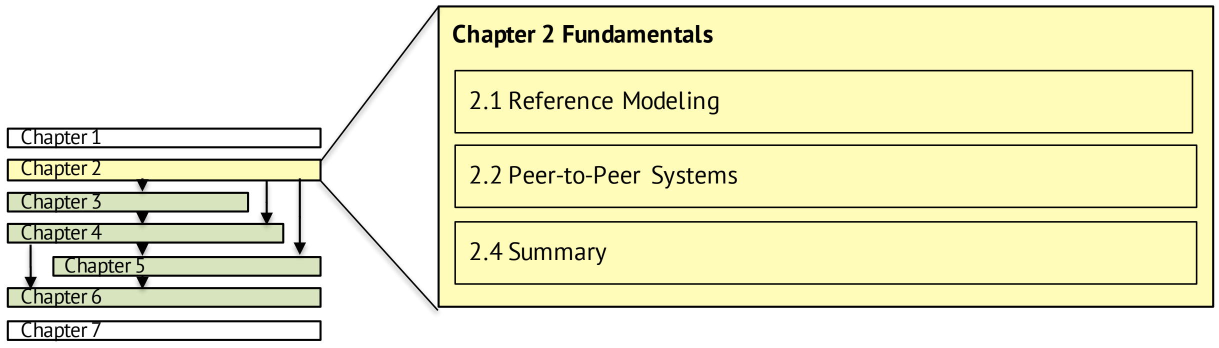


Figure 2.1: Structure of Chapter 2 and its content

It begins with Section 2.1, which introduces the concept of reference models and reference modeling. That provides the theoretical background for modeling the proposed reference model for distributed market spaces, as presented in Chapter 4. Section 2.1.1 reviews reference model definitions and describes their shared characteristics of best practices, universal applicability, and reusability. After, Section 2.1.2 presents different approaches and design principles for reference modeling. They build the foundation for the modeling process and the frame of reference for the reference model for distributed market spaces (cf. Section 4.2).

Section 2.2 introduces the peer-to-peer paradigm essential for understanding the proposed software system architecture for distributed market spaces, which uses the decentralized organization principle; its design, development, and prototypical implementation, as presented in Chapter 5. Section 2.2.1 focuses on the architectures of peer-to-peer systems and their main aspects. Section 2.2.2 introduces the primary components and algorithms essential to enable participants of a peer-to-peer system to join and share information using the decentralized peer-to-peer paradigm.

2.1 Reference Modeling

The term reference modeling first appears in relation to information models, especially data and process models, which were developed for specific domains, but mainly in business informatics. Therefore, reference models are also referred to as reference information models, or conceptual frameworks that represent a domain or a class of domains [FL07].

Some prominent examples are reference models for the supply chain management SCOR model introduced by [Sup05], electronic commerce (e-commerce) [BS04], [Mer02] and electronic business (e-business) by [Lux01], but also the well-known ARIS model introduced by Scheer and Jost [SJ13]. In practice, reference models are used in particular to develop and adapt business and software solutions. However, they are also used in research to examine domains concerning specific requirements of the underlying domain. In this context, their importance is emphasized while at the same time serving as ”theories of business informatics” [Sch13b]. The increasing number of model proposals for different domains led to a deeper reflection on methods for the construction and use of reference models and, thus, the establishment of the reference modeling research area. In the early works presented by Nonnenmacher [Non94], Schütte [Sch98], and Thomas [Tho06], reference models are defined as generally valid and recommendable information models that can be adapted to an application-specific domain according to the configuration principle . In the work by Vom Brocke [Vom03] and later also by Becker, Delfmann, and Knackstedt [BDK07] and Fettke and Loos [FL07], a deeper understanding of reference models is gained. Accordingly, reference models refer to models whose content can be reused to construct other information models. Reuse thus includes the adoption of design results and their adaptation and extension in the application domain.

The vision of reference modeling goes hand in hand with two claims: on the one hand, reference models should serve to adequately describe a domain or class of domains. On the other hand, reference models aim to provide blueprints for a good design of focus systems and the associated organizational settings in that particular domain or class of domains [FL07]. The primary intention of reference models is, therefore, to achieve both effectiveness and efficiency in modeling and developing information systems through reuse [Vom03]. Its direct benefit, therefore, lies in the reduction of modeling and development costs, and thus in time and cost savings, as parts of reference models can be reused and adapted. In addition, reference models formalize recommended practices for a specific domain or application scenario [Tho06], [FL07]. As a result, they provide common models that facilitate learning and a better understanding of the domain under consideration; the focus domain or application scenarios.

2.1.1 Theoretical Background

The research field of reference modeling is relatively young, and there is no consistent understanding and usage of the terminology. This work follows the well-known and widely recognized conceptualization of reference modeling by Fettke and Loos [FL07].

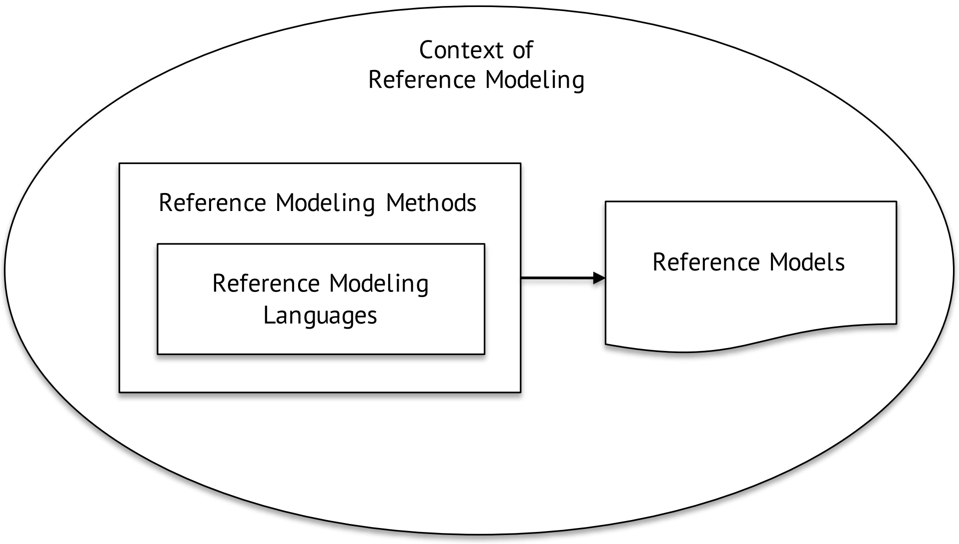


Figure 2.2: Reference modeling framework [FL07]

Accordingly there are four main perspectives that build the theoretical backgrounds of reference modeling as a research field. As illustrated in Figure 2.2, these are:

- *Reference Modeling Languages* – provide constructs, elements and rules in order to support modeling in particular domain or a class of domains.
- *Reference Modeling Methods* – provide methods and procedures by which modeling languages can be used.
- *Context of Reference Modeling* – describes the specific contextual setting (e.g., technical, economic, social, etc.) in which is the modeling process embedded.
- *Reference Models* – are the output of the modeling process representing a domain or a class of domains.

The reference modeling framework by [FL07] generally positions reference models as conceptual frameworks that facilitate the creation of domain-specific conceptual models. Further definitions of reference models go in the same direction describing reference models *"as generic conceptual models that formalize recommended practices for a certain domain"* [Ros03].

Definitions by Misic and Zhao, Schütte, Vom Brocke build on and extend this description adding broader aspects, for example, the system perspective or including the users of the model, the modelers, and other application-related aspects. Misic and Zhao defines reference models *"... as conceptual frameworks for describing system architecture, thus providing a high-level specification for a class of systems"* [MZ99].

Schütte concretize this definition by [MZ99] describing reference models *"... as the result of a construction created by a modeler who declares for IT and business people universal elements and relationships of a system as a recommendation with the help of a language in one point of time so that a point of reference is created"* [Sch98].

The definition by Vom Brocke further distinguishes between modeling activities related to the construction of the model and its application:

"A reference model is an information model that people develop or use for supporting the construction of application models, though the relationship between the reference and application model can be characterized by the fact that the object or content of the reference model is reused by the construction of the object or content of the application model" [Vom03].

Common for all these definitions is, that they describe reference models as conceptual models. But it needs to be noted that not all conceptual models are considered reference models. Likewise, reference models are often simultaneously used with meta-models. However, considering their semantic levels, as recommended by Schütte, reference models need to be differentiated from conceptual and meta-models. While a reference model can be used as a recommendation for constructing models, it is ultimately nothing more than a model; a meta-model, however, describes a modeling language for creating models. Therefore, a reference model considers the semantics of a model and is on the same semantic level as a model. On the other hand, a meta-model is placed one level higher and understood as "a model of a model" that describes the syntax of the model.

In this context, and based on the aforementioned definitions, the literature on reference modeling suggests three main features that characterize reference models. As to [FL07], these are:

- *Best practices* – indicate that a reference model provides best practices for a particular domain. Due to the general structure, most reference models include a set of well-defined best practices for the particular domain. In contrast to the common usage of the term "best-practice solutions", which do offer the best solution for a particular problem, the best-practice character of the reference model indicates that these are rather practice-proven frameworks for the underlying domain [BRS13].
- *Universal applicability* – indicates that a reference model is valid for a domain or class of domains and not for a particular application scenario. By abstracting various practical and theoretical models, reference models provide a wide range of applications, such as domains or entire industries as a class of domains. As such, reference models serve as a starting point for the construction of detailed models, which consider the contextual requirements of the underlying domain or domain classes.

- *Reusability* – denotes that reference models can be understood as blueprints for the development of information systems and thus could be reused in a multitude of different information systems projects. Reusability thus derives from the generality of the reference model since it represents documented knowledge that can be reused in order to create further models or variants of them. Because reference models describe only the general structure for users (i.e., modelers), additional steps are necessary to include their own requirements as well as the contextual requirements of the underlying domain.

In addition to these three primary characteristics, reference models have *to be complete and provide an adequate description of the underlying domain or class of domains*. Such *completeness* of a reference model includes both the semantic and syntactic correctness. This implies that a reference model must not lose its correctness through reusability or adaptability. *Completeness* of a reference model also refers to the completeness of the content; it should contain all the data, activities, elements, or structures required for the modeling in that domain, so that it can effectively support the modeler and ultimately fulfill its purpose of lowering modeling costs. Therefore, completeness is considered a quality feature of a reference model that must be ensured during the modeling process [Sch13a].

Since reference models are considered conceptual models, many authors (e.g., [BRS95], [Moo05], [Sch13b]) recommend using principles for the so-called "appropriate conceptual modeling" [1] introduced by Becker, Rosemann, and Schütte [BRS95]. Even though generic in their nature, these principles should be considered during the modeling process 'by design' and ensure the quality of the intended reference model [BRS95]. [Sch13b] uses these principles in the context of reference models summarizing them into six principles for appropriate reference modeling [2]. These are *constructive fitness, relevance, economics, clarity, comparability,* and *systematic construction*.

The principle of *constructive fitness* has a syntactic and semantic expression. A model is syntactically (i.e., formally) correct if it is complete and consistent with the metamodel on which it is based. The semantic correctness refers to the model's relationship with the object system and denotes its structural and behavioral accuracy. The semantic correctness also includes the intra-model as well as inter-model consistency. The principle of *relevance* has two forms: On the one hand, it must be determined how comprehensive the intended model should describe the underlying domain. On the other hand, it must detail the elements and relationships in a model since they are relevant for using the model effectively. The principle of *economics* refers to the costs of the model. As with *economics*, the *clarity* principle is also highly subjective and individual; it includes aspects like structure, elements, graphic arrangements of information objects, as well as a kind of general legibility. The *comparability* principle refers to the ability of the model to compare to the object system. Finally, the principle of *systematic structure* refers to the using of separate views and integrating the descriptive elements of the

[1] In German: "Grundsätze ordnungsmäßiger Modellierung (GoM)" [BRS95]
[2] In German: "Grundsätze ordnungsmäßiger Referenzmodellierung (GoR)" [Sch13b]

individual views. It also emphasizes using model classes to form a structuring framework for different description views. For a more detailed description of principles for appropriate reference modeling see [Sch13b].

Due to their generic nature, many of these principles are rather abstract and can only indirectly be applied in the context of reference models. For example, ”constructive fitness” can not be interpreted objectively, as they are subject to and depend on the underlying domain. Instead others like ”economics” are subject to the user’s perspective and the practical usage of the model. Since reference models are artifacts that are designed for a specific purpose, the aforementioned principles should be considered during the modeling process, but, they can be evaluated only by applying them in a particular context.

Therefore [AG07], and [Fra07] suggest focusing on a set of principles in respect to the perspective which is considered most relevant for the modeling process. The literature on reference modeling recommends multi-perspective evaluation of reference models as artifacts. Among others, this is the multi-perspective evaluation framework introduced by Frank [Fra07]. It encompasses an economic, deployment, engineering, and epistemological perspective. Each of these perspectives has its own focus, even though they are not independent. The economic perspective is aimed at discussing criteria that are relevant for evaluating costs and benefits, including protecting the possible effects of the investment on information quality and competitiveness. The deployment perspective focuses on criteria that are relevant for the users, stressing criteria such as comprehensibility and compatibility, with other representations being used in an organization (i.e., tools). The engineering perspective aims to evaluate a reference model as a design artifact that has to satisfy a specification that is the claim for general validity for the particular domain. The epistemological perspective aims to evaluate reference models as the results of scientific research. It focuses on criteria for evaluating scientific theories. For more about multi-perspective evaluation methods see work by Ahlemann and Gastl [AG07].

For the purpose of this work, the engineering and deployment perspective are considered essential, and are used for defining quality attributes of the reference model for distributed market spaces (cf. Section 4.1).

2.1.2 Approaches and Design Principles

Reference modeling is considered a process of constructing and applying reference models. As mentioned above (cf. Section 1.1), the reference modeling process includes activities necessary for constructing the intended reference model and activities related to its application in a specific application context. Although different process models exist (e.g., Schuette, Schlagheck, 2000, Frank, Fettke Loos, 2003), the majority of representations follow a cyclical structure that covers the entire modeling process of a reference model [AG07]. In that regard, the reference modeling process is very similar to the widely-used software engineering process. Figure 2.3 illustrates such a general modeling process proposed by [Sch13b].

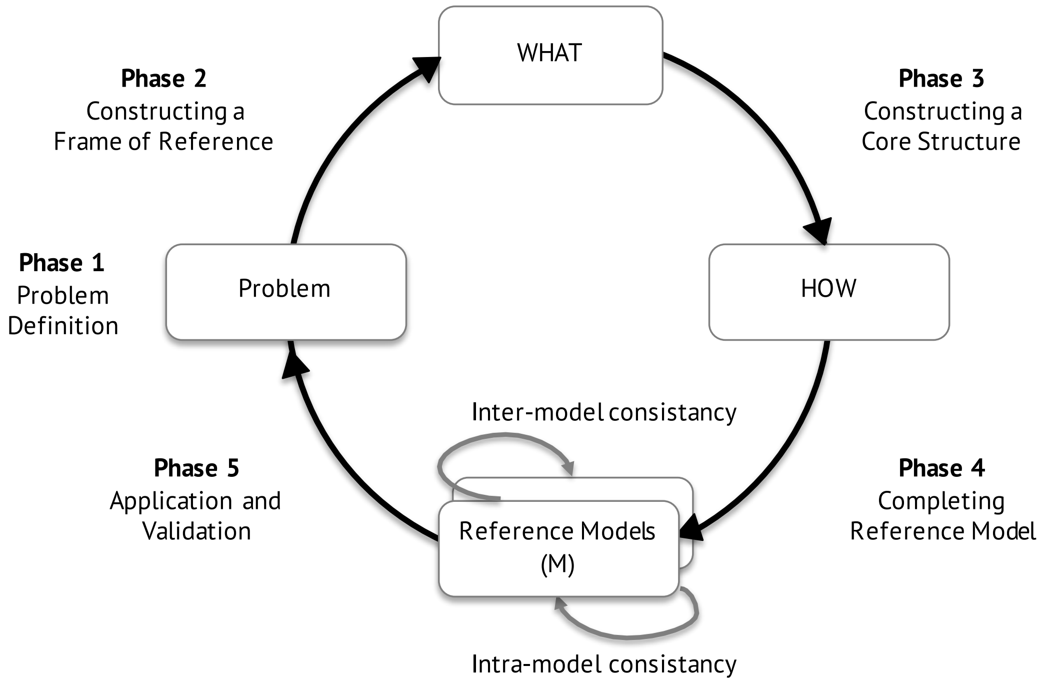


Figure 2.3: Modeling process for reference models [Sch13b]

As illustrated, the modeling process commonly encompasses five phases. As to [Sch98], [Sch13a], [Fra07], [FL04], their main activities can be summarized as following:

- *Phase 1 Problem Definition* – in this phase, the domain for which the reference model is to be developed must be specified. Such a specification of the underlying domain might include a definition of the scope and a brief outline of the intended purposes of the reference model. In addition, the problem definition also can consist of further information relevant for the modeling in this domain, like modeling conventions or different technical terms and details.
- *Phase 2 Constructing a Frame of Reference* – After defining the domain, this phase uses it as the starting point from which to begin the modeling activities. Therefore, the primary aim of this phase is to construct a frame of reference whose primary purpose is to support navigation within the underlying domain. Concerning the following phase 3 (constructing a core structure of the reference model), the purpose of the frame of reference is two-fold: On the one hand, it supports a systematic identification of individual model elements and the selection of modeling language. And on the other, it supports the navigation through the underlying domain. In doing so, the frame of reference guarantees the completeness of the finished reference model [Sch13a].
- *Phase 3 Constructing a Core Structure* – In this phase, the intended reference model is gradually refined based on the frame of reference and the selected modeling language. The output of this phase is a core structure of the reference model

containing all essential model structures, elements, and detailed descriptions of them.

- *Phase 4 Completing Reference Model* – In this phase, the core structure of the reference model is completed. The main focus is on completeness regarding the inter-model and intra-model consistency to ensure the quality of the constructed reference model.
- *Phase 5 Application and Validation* – In this phase, the finished reference model is evaluated in relation to its completeness and consistency by applying it in a specific application context. Furthermore, the finished reference model is evaluated regarding the design goals and requirements, for which it is constructed.

These five phases of the general process model for reference modeling are common to both; the *empirically-founded* and the *configurative* process models. The *empirically-founded* process models (e.g., [AG07], [KPR08]) are based on observing many instances in practice and extracting common elements in reference models. This approach is advisable when many relevant model instances are available for the respective domain. For examples of *empirically-founded* reference models in the domain of platform business models see Section 4.1.

Process models for *configurative* reference modeling (e.g., [Vom03], [Sch13b], [Sch13a]) address the cases when the underlying domain has not been sufficiently researched, but similarities to other reference models can still be drawn. *Configurative* reference modeling is reuse-oriented, and it emphasizes adapting and modifying existing reference models to meet the goals and requirements of the underlying domain.

Another critical aspect of configurative reference modeling considers the adaptation mechanisms and related design principles. As to Becker, Delfmann, and Knackstedt [BDK07], the adaptation mechanisms might be structured in a framework for adaptive support for configurative reference modeling. As illustrated in Figure 2.4, its spans three dimensions. The first dimension (Dimension I) contains the model layer, meta-model layer, and meta-meta model layer. In that, the model layer describes adaptation examples of models. In the representation by [BDK07], these are examples based on EPC (Event Driven Process Chain). The meta-model layer specifies the language of EPCs exemplarily and specifies the modifications, that can be performed within the language during the adaption process, and the meta-meta layer describes the adaptation mechanisms for a particular concept. The literature distinguishes between two fundamental concepts of adaptation mechanisms [BDK07], [BDK04], [Vom03]:

- configurative adaptation
- generic adaptation

Configurative adaptation mechanisms are summarized in the second dimension (Dimension II) of the framework for adaptive support (Figure 2.4). They are based on the principles of ”model projection”. As in [BDK07], model projection describes a process that starts from a ”total model” containing all building blocks and elements for each application context and continues with fading out unnecessary or irrelevant elements for a specific application context. The fading out is realized in a controlled way by

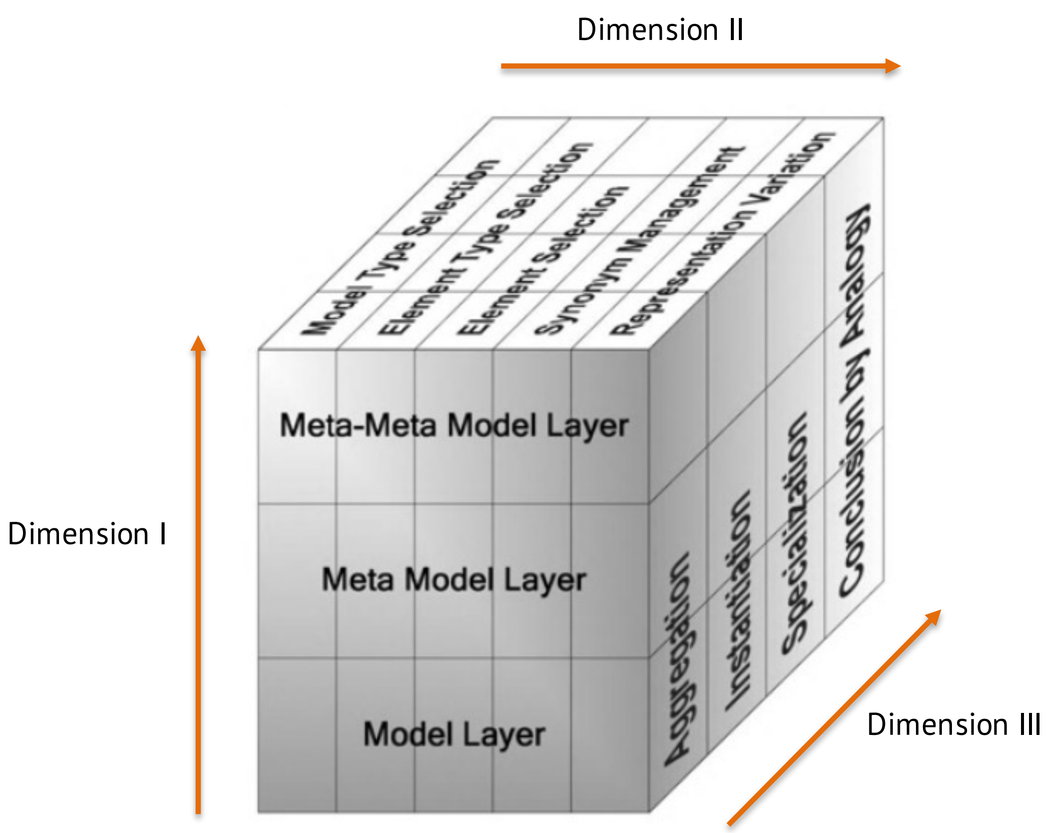


Figure 2.4: Framework for adaptation support for reference modeling [BDK07]

using adaptation parameters as input. Such an adaptation process is also called the ”configuration” as it helps to derive application-specific reference models from the general or meta-models throuhg the process of configuration. It thereby firmly guides the modeling process, offering highly pre-defined paths the model’s users might take during the modeling process. The configurative adaptation mechanisms differentiate between five categories for the configuration [BDK04], [BDK07]:

- Model Type Selection – allows model types in different modeling languages to cover particular needs in a different application context.
- Element Type Selection – allows modeling language variants with different expression power to different user groups.
- Element Selection – enables users to select a specific element, hide or show it in the reference model depending on the application context and unlimitedly users’ requirements.
- Synonym Management – enables users to change element namings to cope with conventions derived from specificities related to different users’ perspectives.
- Representation Variant – enables changing the representation aspect of a modeling language like symbols or model elements.

Table 2.1: Design principles [Bro07], [BDK07]

Principle	Technique
Configuration	Adaptation by selection
Instantiation	Adaptation by embedding
Aggregation	Adaptation by combination
Specialization	Adaptation by revising
Analogy	Adaptation by transfer

Generic adaptation mechanisms are shown in the third dimension of the framework for adaptive support (Dimension III). They introduce additional adaptation principles aimed at enabling more flexibility in the modeling process. The afore-mentioned configurative adaptation mechanisms (Dimension II), consider all model variants incorporated in the reference model, and by doing so they emphasize the principle of "design by selection". In practice, however, situations arise where users' requirements of a model can not be foreseen. In these cases, either the results are inappropriate, or costs of modeling might rise strongly [Bro07]. In contrast, the generic adaptation mechanisms were introduced to emphasize the reuse concept in terms of reusing the content of the model to construct new models. The main purpose of the generic adaptation principles is therefore to give users more flexibility in decision-making during the modeling process. Consequently, using generic principles users must design reference models, rather than designing by selecting, since, as with configuration adaptation, there are no pre-defined model structures.

Moreover, the generic adaptation mechanisms are generally language-independent, but they follow certain rules and principles. In contrast to configurative adaptation, these rules describe the model's relationships from the original to the newly constructed model. However all rules are described by the meta-meta model (see Dimension I, Figure 2.4). The generic adaptation mechanisms encompass adaptation principles *instantiation, aggregation, specialization*, and *analogy*. Together with the above-described *configuration* principle, they build the five main design principles well-established in today's reference modeling (2.1).

In the following, the design principles instantiation and specialization are presented in more detail. They are used in the construction of the reference model for distributed markets spaces, as will be presented in Chapter 4. However, for more a detailed description of other design principles (configuration, aggregation and analogy) as well as their usage, see [Vom03], [Bro07].

Instantiation

> *The principle of instantiation is characterised by the creation of a resulting model "I" by integrating one or several original models "e" into appropriate generic place holders of the original model "G." The resulting model "I," therefore, incorporates the integrated construction results of "e" in "G."* [Bro07]

Instantiation offers the user of a reference model the possibility to specify individual designations and attributes of a model himself when creating a new model variant. When instantiating, reference models only contain generic and general statements about the original model. The design by instantiation requires the working steps illustrated in Figure 2.5.

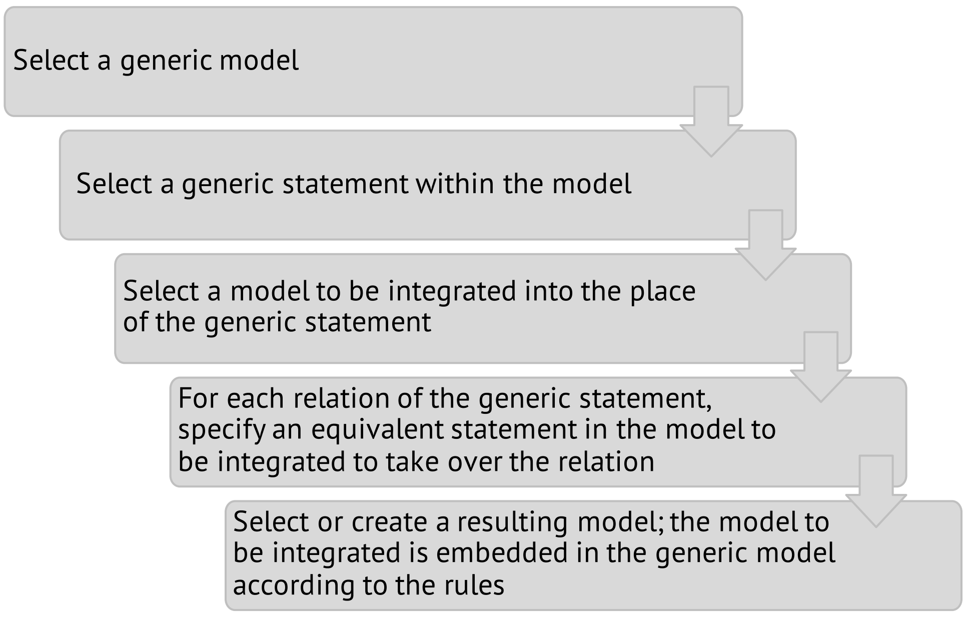

Select a generic model
Select a generic statement within the model
Select a model to be integrated into the place of the generic statement
For each relation of the generic statement, specify an equivalent statement in the model to be integrated to take over the relation
Select or create a resulting model; the model to be integrated is embedded in the generic model according to the rules

Figure 2.5: Design by instantiation – working steps [Bro07]

Specialization

> *Specialization designates deriving a resulting model "S" from a general model "G." That way, all modeling messages in "G" are taken over in "S" and can either be changed or extended. Deleting messages, however, is not provided in the general case.* [Bro07]

Specialization offers the RM user the opportunity to change, expand and adapt all elements of the general reference model. Here, the RM user has unrestricted modeling freedom. The high extent of modeling freedom can lead to inconsistencies and needs to be taken into account. Design by specialization requires completion of the working steps illustrated in Figure 2.6.

The presented design principles configuration, instantiation, aggregation, specialization and analogy can be used individually as well as selectively. Due to the vast variety

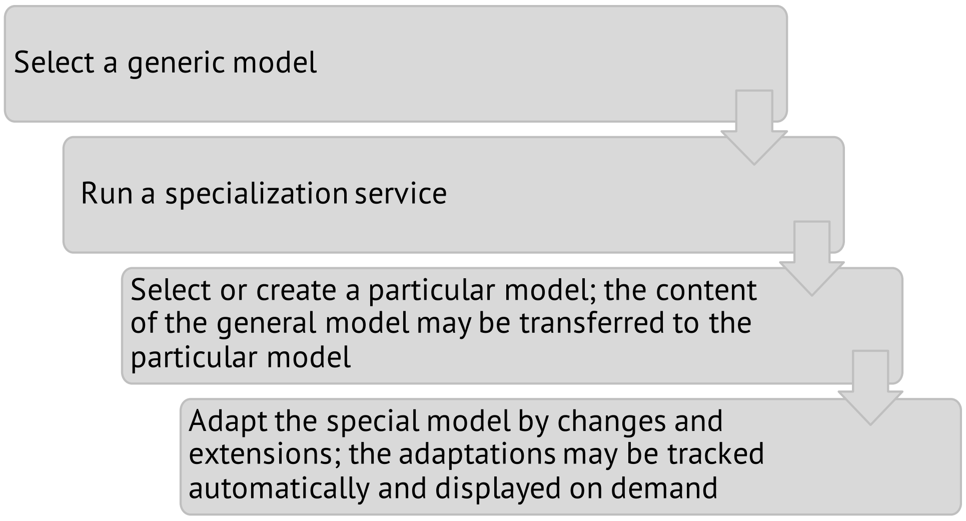

Select a generic model
Run a specialization service
Select or create a particular model; the content of the general model may be transferred to the particular model
Adapt the special model by changes and extensions; the adaptations may be tracked automatically and displayed on demand

Figure 2.6: Design by specialization – working steps [Bro07]

of modeling situations, selective use of multiple design principles in reference modeling seems to be most adequate [Vom03], [BDK07]. Apart from choosing one design principle for a modeling process that appears to be most suitable, combinations of principles can be also aimed at. As to [Bro07] such combinations of design principles can be either model-specific or aspect-specific. The model-specific combination of design principles is advisable when the application domain consists of areas that show different characteristics with respect to the appropriate design principle. This is the case with domains that include a combination of standardized and special processes, which can not be foreseen. Additionally, the aspect-specific combination is suitable in cases when the complexity of the modeling situation might differ with respect to special aspects of the reference model.

2.2 Peer-To-Peer Systems

Peer-to-peer systems essentially work the way we humans interact in real life - we deal directly with one another whenever we wish to [VLO09]. They follow the *peer-to-peer paradigm* for constructing and designing distributed systems and applications with entirely decentralized and self-organized resource usage [CDK05]. As such, they refer to systems that employ distributed resources to perform a function in a decentralized manner [Sch01].

In that respect, peer-to-peer systems are distributed systems without any centralized control or hierarchical organization, where many entities contribute data and computational resources. All entities in peer-to-peer systems, referred to as peers, are autonomous and equal, and participate in the provision of uniform services [Sch01]. Accordingly and as formulated by Steinmetz and Wehrle [SW05], *peer-to-peer systems are systems with completely decentralized self-organization and resource usage.*

The promise of peer-to-peer systems is *better scalability, decentralized coordination of resources*, and *higher fault tolerance* than traditional client/server systems [Mil+02]. Peers in peer-to-peer systems act both as clients and servers and are thus equal peers with symmetric functionality. Furthermore, in peer-to-peer systems, distributed resources are located at the network's edges and close to the peers. They are addressed by the content and not by the server's location, as with client/server systems. To this end, the peer-to-peer systems follow the end-to-end argument [SRC84] which many authors consider as one of the main reasons for the tremendous success of the Internet.

Peer-to-peer systems found their application in many different domains. One of the early age applications of peer-to-peer systems is Napster [Nap01], a platform for content sharing in the music industry. Today's well-known examples are e.g., Bitcoin [Bit11], Etherium [Eth13] for crypto currencies, as well as peer-to-peer marketplaces like [Ope16] and [Baz16] (cf. Section 5.2).

Figure 2.7 shows a peer-to-peer system. The *peers* communicate over end-to-end connections on top of the Internet by forming an *overlay network*. A *peer-to-peer overlay network* is a virtual network composed of different connections between peers on top of an existing network infrastructure (i.e., Internet) [SW05], [VLO09].

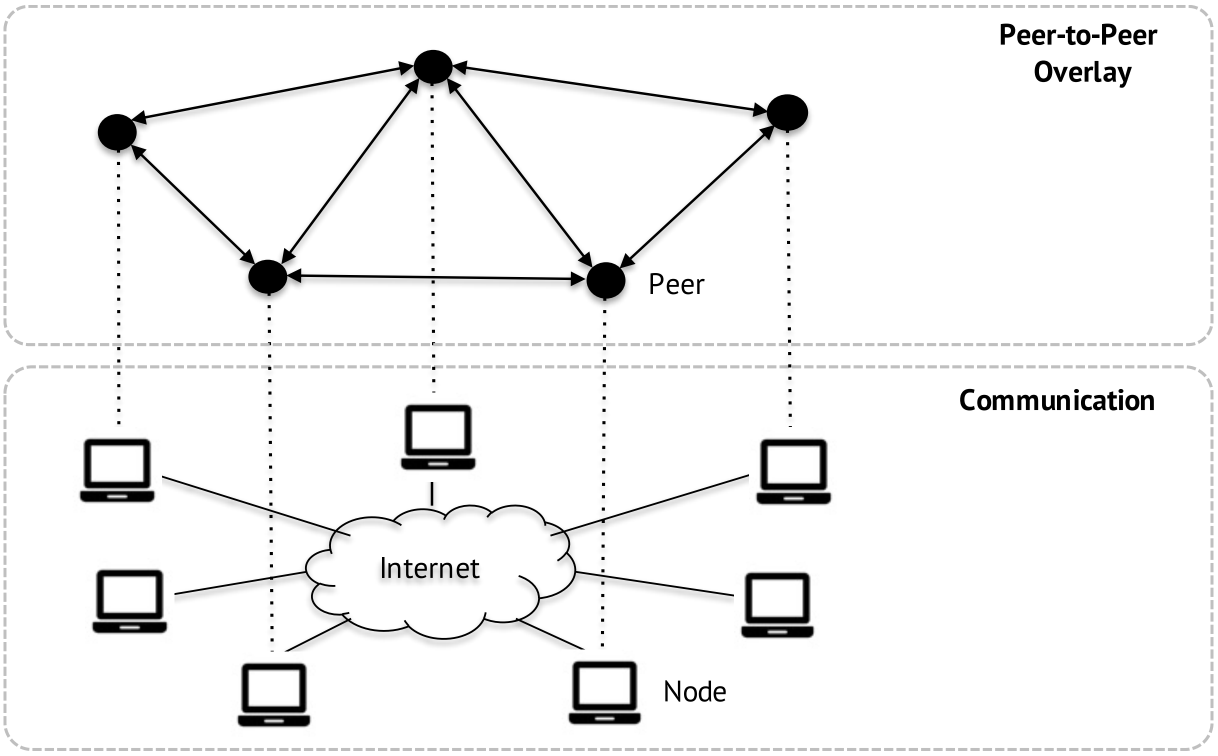


Figure 2.7: Peer-to-peer system (based on [VLO09])

As a form of distributed systems, peer-to-peer systems share the core characteristics of distributed systems, such as fault tolerance, the symmetric role of participants, and scalability. However, they also distinguish themselves from traditional distributed systems regarding heterogeneity, dynamism, and distributed control. Peer-to-peer systems can be very heterogeneous in terms of the capacity of a particular peer (i.e., node). They often work in a highly dynamic environment due to joining new nodes or leaving existing nodes. In this regard, peer-to-peer systems emphasize self-organization, adaptation, and resilience, and thus advocate for autonomicity in systems in general [VLO09].

The following three main characteristics are the primary attributes of a peer-to-peer system [CDK05], [VLO09]:

- *Decentralization* – peer-to-peer systems are based on the assumption that no centralized components are required. This means that their correct operation does not depend on the existence of any centrally administered components.
- *Equality* – All peers are equal, as they have the same functional capabilities and responsibilities. Although they may differ in the resources that they contribute, all the nodes in a peer-to-peer system contribute resources to the system.
- *Autonomy* – Peers are autonomous entities. They are owned by individuals who are fully autonomous regarding their respective resources.

2.2.1 Architecture of Peer-to-Peer Systems

Peer-to-peer systems share the general characteristics of decentralization, equality, and anonymity, but their design differs considering two fundamental architectural dimensions:

- degree of the centralization, and
- structure of the overlay network

The degree of the centralization refers to the availability of one or more servers, and to what extent the peers depend on the services provided by those servers. Regarding the implemented degree of decentralization, the architectures of peer-to-peer systems are generally divided up into three categories [VLO09], [Mil+02]. As shown in Figure 2.8, these are *centralized peer-to-peer systems, decentralized peer-to-peer systems*, and *hybrid peer-to-peer systems*.

Centralized peer-to-peer systems

Centralized peer-to-peer systems combine the features of both centralized (e.g., client/server) and decentralized architectures. As in a client-server system, there are one or more central servers (see Figure 2.8, left) which maintain metadata of resources shared by peers, support peers to locate desired resources, and serve as task scheduler to coordinate actions among peers. To locate resources, a peer sends messages to the central server to determine the addresses of peers that contain required resources. Once a peer has the necessary information, it can communicate directly with other peers. The previously mentioned Napster [Nap01] is realized as a centralized peer-to-peer system, whereby the file exchange was realized via peer-to-peer (like file search and lookup) as client/server.

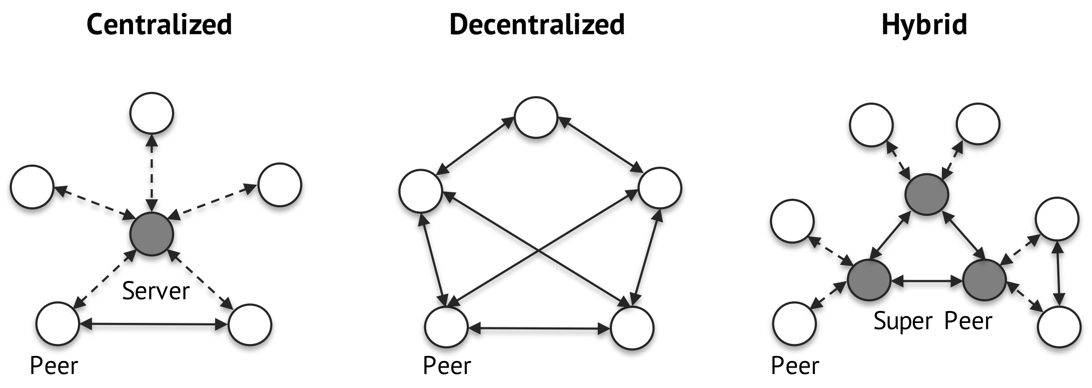


Figure 2.8: Categories of peer-to-peer systems

Centralized peer-to-peer systems have two main advantages: On the one hand, they speed up the process of resource location and guarantee to find all possible peers that maintain the desired resources, on the other hand, they are relatively good to organize, maintain and keep consistent through the one or more central servers. However, as in all systems that rely on central components, centralized peer-to-peer systems are predisposed to the single point of failure and malicious attacks. Furthermore, the server as a centralized component can become a bottleneck for a large number of peers, potentially diminishing performance, scalability, and robustness (cf. [SW05], [VLO09], [CDK05]).

Decentralized peer-to-peer systems

In decentralized peer-to-peer systems, each peer has equal responsibilities and rights, and there are no centralized components to coordinate the operations among peers. These are considered as systems in their purest form, as all relations are peer-to-peer (see 2.8). As such, decentralized peer-to-peer systems avoid a single point of failure since all tasks and services are distributed throughout the network, and no peer is indispensable to the system. As a result, the network has ”strong immunity” to censorship, partial network technical failures, and malicious attacks. Thereby malicious attacks refer to the malicious behavior of peers, which might deliberately act maliciously to harm the system (e.g., by forwarding wrong requests and fake requests or query responses).

Hybrid peer-to-peer systems

Hybrid peer-to-peer systems combine centralized and decentralized approaches to leverage the advantages of both architectures. They maintain scalability, similar to decentralized peer-to-peer systems, and there are no servers as central components in hybrid peer-to-peer systems. As shown in Figure 2.8, there are peers possessing much more powerful capabilities and having more responsibilities than other peers, which are referred to as ”super peers”. These super-peers form an ”upper layer” of a peer-to-peer hybrid system, which provides similar services for the ordinary peers as the central server does in a centralized peer-to-peer system. The common peers, on the other hand, can use much more services from the super peers in the ”upper layer”, especially in the process of resource location. Nowadays, many peer-to-peer applications employ a hybrid peer-to-peer architecture such as BitTorrent [Bit], a widely-used protocol for distributing data and large files over the Internet. Hybrid peer-to-peer systems are more efficient than fully decentralized peer-to-peer systems regarding resource location and performance. However, they can have different strengths and weaknesses depending on the particular application domain and its requirements concerning scalability, load-balancing, autonomy, and anonymity.

The structure of the overlay network concerns the logical network topology in terms of whether it is *structured* or *unstructured*. As explained above, an overlay network is a virtual network composed of direct connections between peers on top of existing network infrastructures, e.g., the Internet (cf. Section 2.7). Peers communicate over end-to-end connections, thereby forming an overlay network. Peer-to-peer systems with an unstructured network topology are called *unstructured peer-to-peer networks* and those with a structured network topology *structured peer-to-peer networks*.

Unstructured peer-to-peer networks

In *unstructured peer-to-peer networks*, peers randomly connect to other peers, and there is no relation of the placement of the resources (i.e., data objects) with the network topology. Consequently, peers have no or limited information about data objects stored by other peers; thus, searching in unstructured peer-to-peer networks may query all neighbors for data items that match the query (i.e., flooding). The main difference between unstructured and structured networks lies in how queries are being forwarded to other peers in the network. In an unstructured peer-to-peer system, each peer is responsible for its data objects and keeps track of a set of neighbors that it may forward queries to. There is no strict mapping between the identifiers of data objects and those of peers. As a result, locating data in such a system is challenging since it is difficult to predict which peers maintain the queried data precisely. Furthermore, there is no guarantee of the completeness of answers, and there is no guarantee of response time (unless the entire network is searched).

Structured peer-to-peer networks

In *structured peer-to-peer networks*, data placement is under the control of certain predefined strategies, e.g., a distributed hash table (DHT) that enables mapping between data and peers. More importantly, these networks guarantee (precise or probabilistic) search cost. However, this is typically at the expense of maintaining certain additional information. Employing the principle of the mapping, most of the structured peer-to-peer networks, including, e.g., Chord [Sto+01], Tapestry [Zha+04], Pastry [RD01] adopt the key-based routing strategy to locate the desired resource. As a result, a request can be routed to the peer who maintains the desired data quickly and accurately. They all guarantee that the peer distance and the number of connections per peer are $O(\log n)$, where n is the number of peers in the network. However, since the placement of data is tightly controlled, the cost of maintaining the structured topology is high, especially in a dynamic network environment, where peers may join and leave the network at will.

2.2.2 Components and Algorithms

While peer-to-peer systems can be architectured differently, they share many common core components required to build and operate a peer-to-peer system. Figure 2.9 gives an overview of these components according to the classification by [Mil+02]. The first three layers include the basic components necessary for every peer-to-peer system. The latter two include components that are considered necessary for a specific class of peer-to-peer systems or a specific type of application.

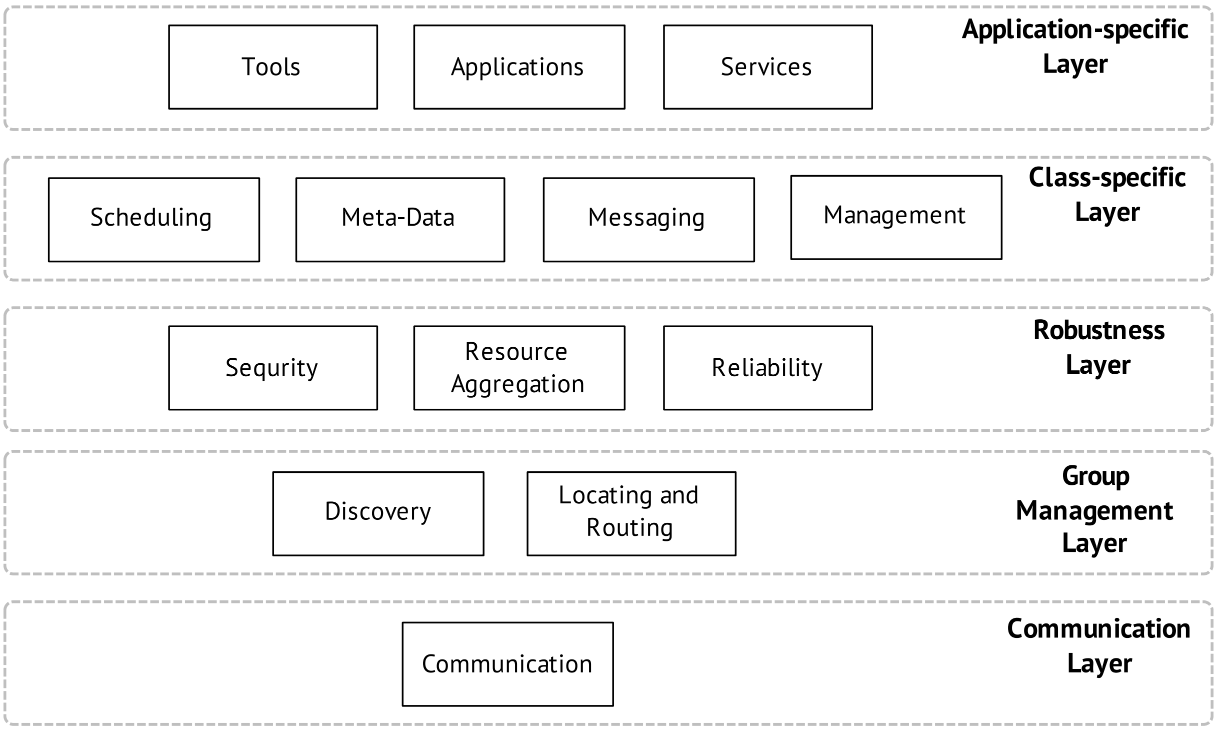


Figure 2.9: Components of a peer-to-peer system [Mil+02]

The most common application types in peer-to-peer systems summarized by Milojicic et al. [Mil+02] are parallelizable applications, content and file management applications, and collaborative applications. Parallelizable applications focus on large compute-intensive tasks splitting them into smaller parts that can be executed in parallel over many independent peer nodes. Content and file management applications focus on storing and retrieving information among the network. Collaborative applications focus on supporting collaboration in real-time, without relying on a central instance to collect information.

The following briefly introduces each of these five layers (for a detaled description see [Mil+02],[SW05],[VLO09]):

- *Communication Layer* includes a *communication* component that covers a wider spectrum of communication paradigms; from desktop mashines, over other devices connected to the Internet to sensor-based devices which are connected in an ad-hoc manner via a wireless medium. The main responsibility of these components is to maintain connectivity in a dynamic environment of a peer-to-peer system.
- *Group Management Layer* includes *discovery* and *locating and routing* components responsible for discovering other peers in the peer-to-peer system, and location and routing between those peers. The discovery algorithms are subject to many factors (e.g., type of the device) and can be designed as decentralized or centralized solutions based on centralized directories (e.g., Napster). Location and routing algorithms generally try to optimize the path of a message traveling from one peer to another and thus minimize the hops.
- *Robustness Layer* includes components necessary to maintain a robust peer-to-peer system. *Security* component is responsible for ensuring that only trusted or authenticated sources should have access to information and services provided by a given node. *Resource Aggregation* component is responsible for aggregating resources available within the system, as this is the basis for interacting with peers. *Reliability* component is necessary to guarantee reliable behavior within a peer-to-peer network as a dynamic environment. It is an inherent challenge of a peer-to-peer network due to its distributed nature; the most common solutions take advantage of redundancy. For example, in the case of a messaging application, lost messages can be resent or sent along multiple paths simultaneously.
- *Class-specific Layer* encompasses components that are specific for a class of peer-to-peer systems. They abstract functionality for that particular class. *Scheduling* component, for example, applies to parallelizable or compute-intensive applications, where compute-intensive components are broken into pieces that must be scheduled across the network. *Meta-Data* applies to content and file-management applications and is responsible for describing the content stored across the nodes and determining the location of the desired information. *Messaging* and component *Management* apply to collaborative applications, as they enable communication between peers and support managing the underlying network infrastructure.
- *Application-specific Layer* encompasses different *tools, applications*, and *services* necessary to implement application-specific functionality. These are considered the constitutive parts of the application functionality, that is, applications running on the underlying peer-to-peer network.

[Mil+02] describes the three common algorithms used in peer-to-peer systems *centralized directory model*, *flooded request model* and *document routing model* as follows.

In the *centralized directory model* peers connect to a central directory where they publish information about the content they offer for sharing. Such a central index matches the requests with the best peer that can meet the request. Depending on the peers' needs, the best match can be the cheapest, fastest, or most available. The file exchange occurs directly between the two matched peers. The centralized directory model requires a directory server, which hosts information about all participants in the network. This model is applied in centralized peer-to-peer systems (cf. Section 2.2.1) and is considered efficient despite certain scalability issues.

In the *flooded request model* each request from a peer is flooded (i.e., broadcasted) to directly connected peers, which themselves flood their peers until the request is answered or a maximum number of flooding steps occur, which is typically 5 to 9 steps or hops. This model is considered "a true peer-to-peer" and is applied in decentralized peer-to-peer architectures. The main issue of this model is scalability, but there are different approaches and methods to improve it. Caching recent search requests and using software to concentrate requests is also used to improve scalability.

In the *document routing model* each peer from the network is assigned a random identifier (ID), and each peer also knows a given number of peers. When a document is published in the system, an ID is assigned to the document. This ID is based on a hash value of the content and the name of the document. Each peer will then route the document towards the peer with the ID that is most similar to the document ID, whereby the process is repeated until the nearest peer ID is the current peer's ID, and a local copy of the document is kept. When a peer requests the document, the request will go to the peer with the ID most similar to the document ID. This process is repeated until a copy of the document is found.

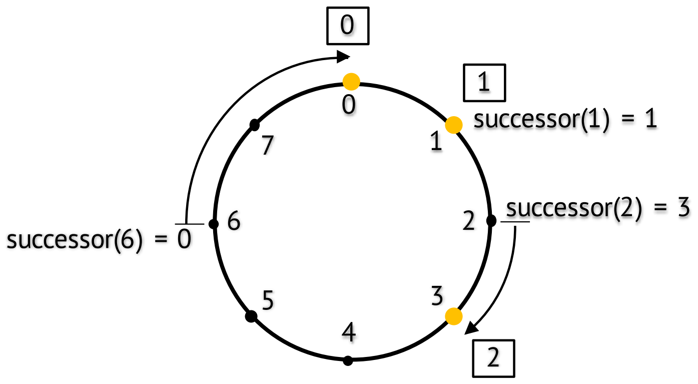
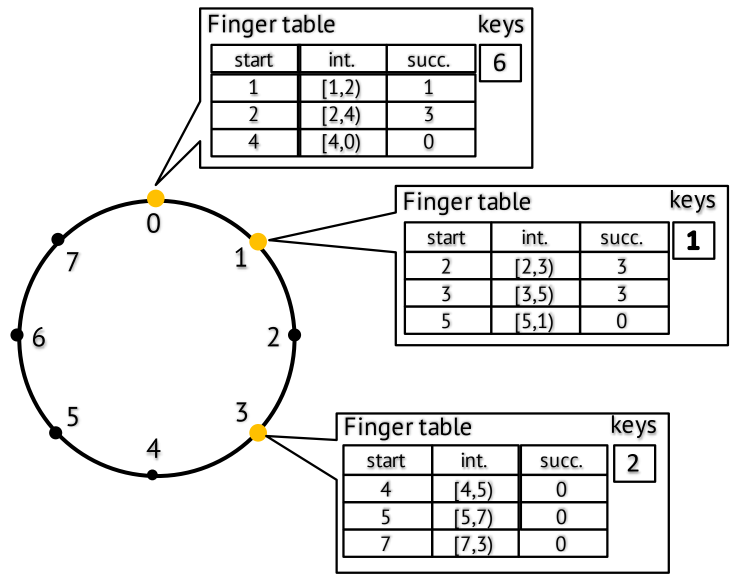


Figure 2.10: Identifier ring consisting of nodes 0, 1, and 3 [Sto+01]

The most structured overlay networks (see Section 2.2.1) e.g., Chord [Sto+01], Tapestry [Zha+04], Pastry [RD01] implements the document routing model. As an example of structured overlay networks, the Chord is shortly presented in the following. While other structured overlays work in a similar way, the Chord is chosen since it is used for the prototypical implementation of the architecture for distributed market spaces presented in Chapter 5 and its demonstration in a testbed environment as presented in Chapter 6.

Chord is a protocol that uses DHT for efficiently locating the peer that stores the data item corresponding to a given one-dimensional key value. It implements the document routing algorithm modeling the identifier space as a uni-dimensional, circular identifier space. Chord uses consistent hashing [Kar+97] to map the domain of search keys uniformly into the Chord domain of keys. Each peer is identified by an m-bit key value generated by the hash function applied on the peer's IP address. The peers are ordered in an identifier circle based on their key values.



start	int.	succ.
1	[1,2)	1
2	[2,4)	3
4	[4,0)	0

start	int.	succ.
2	[2,3)	3
3	[3,5)	3
5	[5,1)	0

start	int.	succ.
4	[4,5)	0
5	[5,7)	0
7	[7,3)	0

Figure 2.11: Finger tables and key locations for nodes 0, 1, and 3 [Sto+01]

Each object with a key-value "k" is assigned to the first peer with a key-value equal to or larger than "k". Figure 2.10 shows an identifier ring consisting of nodes 0, 1, and 3. To maintain effective communication between peers, every peer stores the addresses of its predecessor and successor on the identifier circle. In addition, each peer maintains a routing table, the so-called "finger table" with addresses of up to "m" other nodes. Figure 2.11 exemplary presents the finger tables and key locations for nodes 0, 1, and 3. With high probability, the peer responsible for a given key value is located via *O(log(N))* number of messages to other peers. For a more detailed description of the Chord protocol, see [Sto+01].

2.3 Summary

This chapter laid the theoretical background for this work. It gave an overview of reference models and described the reference modeling process used to build the reference model for distributed market spaces (cf. Chapter 4). The common characteristics of best practices, universal applicability, and reusability of reference models are described in detail, and different approaches and design principles for reference modeling are presented. Additionally, this chapter introduced the peer-to-peer systems that follow the decentralized organizational paradigm. It outlined their primary components and algorithms used to design, develop, and prototype the architecture for distributed market spaces (cf. Chapter 5 and Chapter 6).

3 | Distributed Market Spaces as a new Concept for the Post-Platform Economy

This chapter introduces *distributed market spaces* as a new concept to facilitate the emergence of the post-platform economy. It outlines the background of the concept by examining its primary drivers and characteristics. Building on this, Chapter 3 identifies the overall objectives of distributed market spaces, defining the scope of the domain for which a reference model is constructed and presented in Chapter 4. The overall objectives also justify the requirements for the underlying software system architecture, as introduced in Chapter 5.

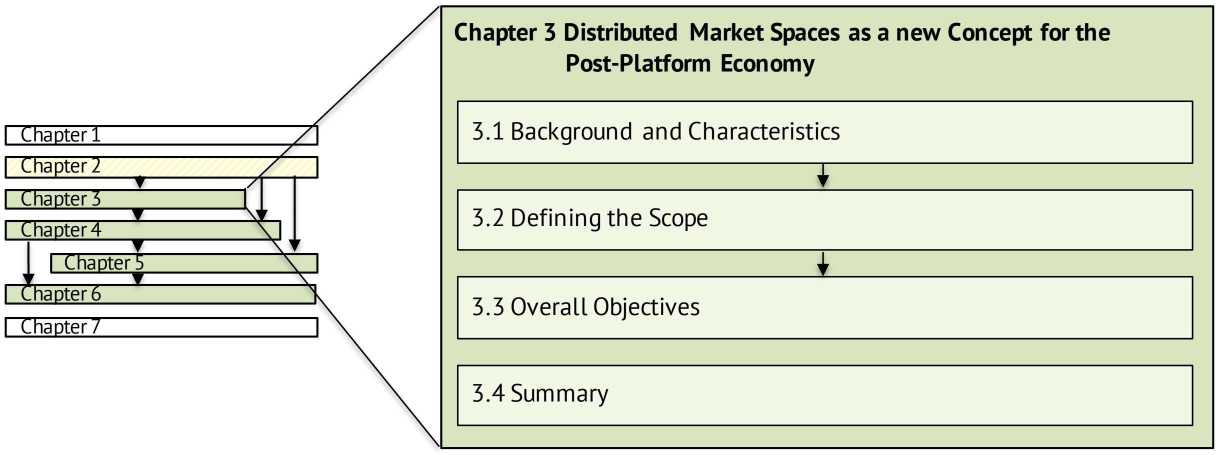


Figure 3.1: Structure of Chapter 3, its content and thematic dependencies

The primary purpose of distributed market spaces is to alleviate the adverse effects of an increasing platform power and lower transaction costs for complex products while maintaining the benefits and enabling nature of contemporary platform models. The key drivers of the concept are, therefore, decentralization and novel consumer orientation – with novel consumer orientation emphasizing personalized consumers' needs to be represented by the market exchange of complex products.

Figure 3.1 visualizes the structure of this chapter. It begins with Section 3.1, which examines the background, the key drivers of decentralization, and complex products. It identifies the main characteristics of distributed market spaces as decentralized and self-organized organizational structures. Afterward, Section 3.2 defines the scope and analyzes the domain of distributed market spaces in terms of business, organization, architecture, and technology aspects. Section 3.3 summarizes the analysis findings, addressing their implications for designing and developing a reference model. Thus it identifies the overall objectives for reference modelling such a domain. Chapter 3 closes with a summary in Section 3.4.

The essential parts of this chapter are presented and published in [RP15], [PRR16], and [RWP17].

3.1 Background and Characteristics

As previously defined in Section 1.1, the post-platform economy refers to a range of economic activities enabled by self-organized and decentralized online structures. The primary purpose of these structures, defined as distributed market spaces, is to assess the adverse effects of increasing platform power and lower transaction costs for complex products while retaining the advantages of platform models. Given this, the key drivers of distributed market spaces are *decentralization and novel consumer orientation*. In this, *novel consumer orientation* emphasizes the personalized consumer needs represented by the market exchange of complex products. The following examines these two drivers in more depth and outlines the elementary characteristics of distributed market spaces.

3.1.1 Decentralization as a Driver

The core of the platform business model is the creation of value by providing an open, supporting infrastructure that enables consumers and providers to plug in to interact and transact with each other [PVC16], [ES16]. Therefore, the main aim of platform design is to create a supportive infrastructure to enable the essential activity that takes place on a platform – the core interaction. Core interaction consists of a set of simple, repeatable actions that providers and consumers take to create and consume value through the platform [MJ16], [PVC16]. On the one hand, core interaction enables the exchange of value that attracts the most participants to the platform [PVC16]. On the other hand, it facilitates the creation of a network of participants, necessary for establishing desired network effects. Hence it enables the platform to turn that network into the value [PVC16], [CGY19].

The core interaction is realized by a set of supportive functions. Together they build the infrastructure of a platform as illustrated in Figure 3.2, on the left. That is the reason why the primary aim of platform design is to create an infrastructure, which compounds systems that realize the necessary functions as powerfully and effectively as possible [PVC16],[RR17]. To emphasize the complexity and dependencies of such an attempt, Moazed and Johnson in [MJ16] compares a well-designed platform infrastructure with a human hand. A platform as a ”visible hand” with the core interaction via the thumb and supportive functions using the fingers. As with a human hand, it needs to function as a whole... ”without the thumb, the other fingers are not very useful, but also without a finger, the whole hand does not function as well.”

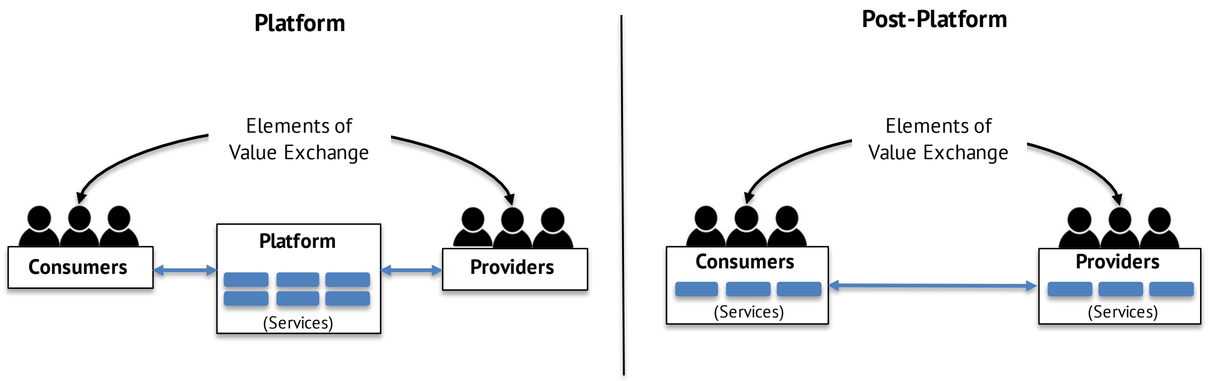


Figure 3.2: Platform economy vs. post-platform economy

Different platforms realize different supportive functions, which strongly depends on how the core interaction is created on a particular platform. However, generally, supportive functions promote aspects of decision-making, information processing, and network governance built around the platform’s infrastructure. As mentioned above, the literature on platform business models considers four functions as the key supportive functions of a successful platform [ES16],[PVC16],[RR17]. To recall, these are:

- Attracting the audience
- Matchmaking
- Providing tools and services, and
- Establishing rules and standards to govern the network of participants and, thus, build trust and maintain quality.

Attracting the audience refers to activities, features, and processes by which a platform is enabled to attract a critical mass of consumers and providers. It encompasses the attraction and subsequent management of the network of participants. Without a platform, it could fail. Hence, the primary purpose of each platform model is to grow the network and create potential connections that might turn into interactions and thus lead to desired network effects.

Matchmaking refers to the capability of a platform to connect the right consumer with the right producers in order to facilitate value exchanges among them. Thus the main aim is to create the most effective and scalable matchmaking function that is considered critical to the success of the platform. For the matching to be effective, results must meet participants’ needs by being relevant, valid, timely, and presenting the right amount as well as the depth of information.

Providing tools and services refers to the capability of a platform to support its participants to exchange value in a way that lowers transaction and coordination costs, and in this way, makes the platform more valuable over time. Tools and services should support participants in creating more value from their value exchanges. In contrast to the previous two functions (attracting and matchmaking), tools and services direct at a particular step in the core interaction and are held to play one of the crucial roles in consumer satisfaction.

Establishing rules and norms refers to the ability of a platform to govern the network of participants in a way to create value fairly distributed among all those who add value. Therefore rules set guidelines that govern which behaviors are allowed and encouraged and which are forbidden or discouraged. On the other hand, norms concern the question who gets access to the platform, how to divide the value generated by the platform, and how to resolve conflicts among participants.

The core interaction and supportive functions are under the direct control of platform owners. They build a collection of mechanisms through which they influence and exercise control over the platforms' participants [Tiw13]. On the one hand, this is one of the main enablers of platform business models. As a result, more and more economic activities in the platform economy are orchestrated through centrally created and managed networks [MJ16],[CGY19]. However, and as pointed out above (cf. Section 1.1), this underpins the positional power of the platform, putting them in a position of a monopoly, where they can dictate the rules, control access, and offerings, and thus lead to a de-facto centralization of previously decentralized offerings on the Internet.

As defined in this thesis, the concept of the post-platform economy aims to alleviate such effects and no longer rely on centrally organized platforms as the model to facilitate market exchange. It relies much more on distributed market spaces, shifting the paradigm of contemporary platforms towards decentralization and moving in the direction where core interaction and supportive functions are realized in a decentralized manner. This is illustrated in Figure 3.2, on the right.

This shift has two implications: Firstly, it implies that the core interaction (the essential activity that takes place on a platform) needs to be agreed upon and committed upon by the constitutive parts of a distributed market space, that is, by its participants. Secondly, it implies that the supportive functions need to be provided by the market participants themselves, which needs to self-organize to ensure that necessary functions are provided collaboratively and reliably. Consequently, the concept of distributed market spaces needs to direct towards decentralized decision-making, information processing, and governance.

Therefore, *distributed market spaces are characterized as strictly decentralized and self-organized online structures*:

- *Strictly decentralized* refers to the decentralized realization of the supportive infrastructure of distributed market spaces. It addresses different aspects of decision-making, information processing, and data management necessary for the provision of core functions decentrally.
- *Self-organized* refers to aspects of distributed market spaces as an organizational structure with decentralized governance. It emphasizes its ability to empower

participants to establish and govern a market space on their own, where they can exchange value directly and reliably. Beyond that, it emphasizes participants' ability to agree upon their own rules and norms and, by doing so, to govern and uphold a self-organized exchange environment.

3.1.2 Complex Products as a Driver

The use case of a couple that plans an evening with friends introduced in Section 1.1 represents an example of a class of real-life situations consumers face when looking to satisfy highly personalized demands. As a consumer, our couple from this use case requires a bundle of different services, including buying theatre tickets, a dinner table reservation at an Italian restaurant, a parking spot, and a babysitter. To be satisfied, such a bundle of desired services needs to fulfill a particular context. That particular context describes conditions, requirements, and constraints defined by our couple as a potential consumer. For example, this might include the desired time scope, location of the theater and surrounding parking options, as well as personal preferences considering the genre of the play (e.g., drama, comedy), cusine of the restaurant (e.g., Italian), and a well-rated and reliable babysitter.

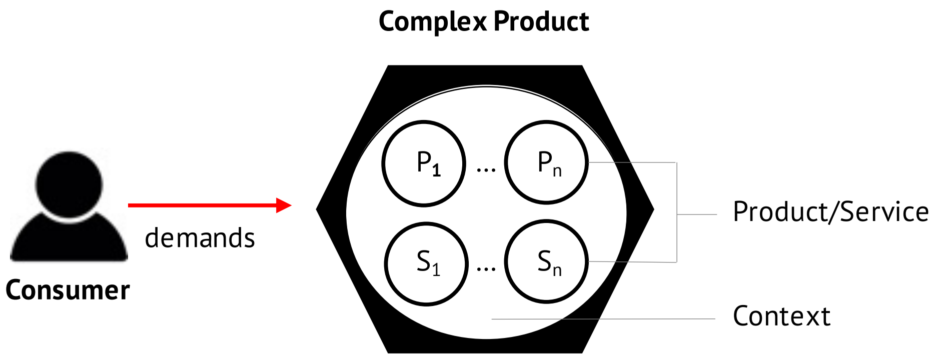


Figure 3.3: Conceptual structure of a complex product

Another application scenario going in a similar direction considers the owner of a house who wants to renovate his bathroom. Besides a lot of thinking and planning, this endeavor also includes ordering specific products like new tiles, sanitary facilities like lavatory, shower, and hiring installers for plumbing and heating. As with the previous case, the bundle of products and services demanded from the house owner also has to meet a set of requirements, for example, to fit the budget, schedule, and quality constraints. That is, to be personalized in a way to address the demand related to the given bathroom conditions. The use cases above represent fundamentally different intentions, but they are very similar from the consumption point of view. Our couple and the house owner demand an arbitrary combination of individual products and services, which, as a combination, has to meet a set of requirements and constraints. Thus they order complex products to satisfy a personalized need. Note that these are just two examples of complex products from business-to-consumer or consumer-to-consumer backgrounds. The same applies to complex products from the business-to-business context, including all commercially tradable products and services. For example, complex products from the energy sector, intending to personalize the energy supply for both private and business consumers. Furthermore, diverse Internet of Things services are included in the different Smart City and Smart Region initiatives.

Figure 3.3 visualizes the conceptual structure of a complex product. A complex product is initiated by a consumer, who demands it by formulating a particular request for a combination of individual products (P_1, ..., P_n) and services (S_1, ..., S_n). An individual product (P) or service (S) represents any electronically tradable object. Tradable objects, as to [Gre15], can be categorized into four main categories: *physical products, digital products, services* and *financial products*. *Physical* products are tangible goods usually exchanged on a per-piece basis like, for example, theatre tickets, tiles, or lavatories. *Digital products* are intangible goods that are exchanged electronically like, e.g., restaurant reservations. *Services* are regarded as activities that one market participant performs for another, e.g., babysitting or plumbing-related activities. *Financial* products are considered amounts of money or other assets transferred between consumers and providers in a particular context (e.g., insurance). Tradable objects can also involve bartering, representing exchanges where non-financial assets have been exchanged both ways.

The demanded combination of individual products (P_1, ..., P_n) and services (S_1, ..., S_n) has to fulfill a particular *context*. Context represents a broader range of information, which personalizes the demand and adds synergetic value to the required product and service combination. The synergistic value expresses an overall value to the consumer greater than the sum of the value of individual products and services. The consumer-defined context, as used in this work, follows the definition provided by [Dey01]. Accordingly, "... the context is any information that can be used to characterize the situation of an entity, which is considered relevant to the interaction between a user and an application, including the user and the application themselves". In that sense, a combination of individual products and services is an entity. The context is the information that a consumer acknowledges as relevant for this entity to meet a personal demand.

As illustrated by use cases above, the context of a complex product might encompass much more information than related to the required products or services (i.e., product or service description or composition possibilities). For example, it can contain decision criteria (i.e., different constraints, like price, configurations, payment, and delivery modalities) that can affect the complex product as a whole and include a more extensive range of relevant data. Among others, these can combine data related to the consumer's requirements such as time, place, temperature, personal preferences like ”regional” or ”fair-trade products”, or reviews and recommendations by trusted social contacts and online communities regarded other matters, as in case of, e.g., hiring a babysitter. The required context information can be integrated manually by the consumers, derived from data delivered by interconnected IT systems, or obtained from sensors and actuator networks providing, e.g., weather or other environmental data relevant for a specific complex product.

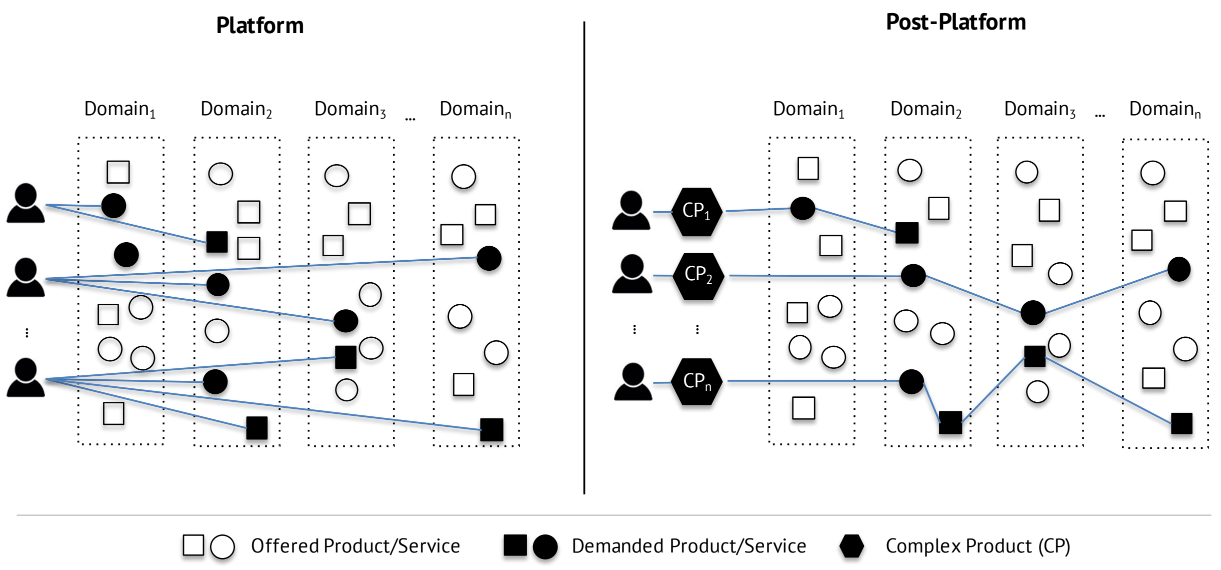


Figure 3.4: Complex products in platform vs. post-platform economy

Complex products are already possible today, but require a great deal of manual effort for the consumer. The modern platform economy is vertically oriented and focused on the availability and exchange of individual products and services. Consequently, the more complex the demanded product/service combination, the more effort consumers must invest in making an informed decision (see Figure 3.4). As illustrated on the left, consumers first need to know where and how to find viable offers for required products and services, then aggregate all relevant information manually and put them in the context of given requirements and constraints. This complexity of finding an optimal product/service combination, which usually might span different platforms, can easily overwhelm consumers, leading to the decision-making following the principle of choosing what is ’good enough’ instead of the ’optimal’ (i.e., adverse selection [Ant16]). Moreover, this contradicts the primary purpose of platforms as matchmakers [ES16], that efficiently connect the right consumers with the right providers and, by doing so, significantly reduce transaction costs and thus facilitate market exchange.

To support complex products, the concept of distributed market spaces needs to shift the vertical orientation of the platform economy towards consumers and their increasingly horizontally oriented complex needs. Figure 3.4 indicates this shift from individual products and services in the platform economy towards complex products $(CP_1, \dots, CP_n)$, as shown on the right. Consequently, the third primary characteristic of distributed market spaces is the ability:

- to empower consumers to formulate their complex demands in the form of complex products, and based on that,
- to support the market exchange of complex products

in the same effective way as platforms support individual products and services today.

3.2 Defining the Scope

The scope of distributed market spaces is defined by the primary characteristics of the underlying domain. As previously presented, these are decentralization, self-organization, and the ability to effectively support the market exchange of complex products as platforms do for individual products and services. Each of these characteristics imposes specific requirements on distributed market spaces as an organizational model and, ultimately, reference modeling in this domain. Therefore, this section focuses on analyzing further aspects of the underlying domain, particularly those related to complex product scenarios.

The analysis of complex product scenarios follows the recommendation of the BOAT method [Gre15]. The acronym BOAT stands for business (B), organization (O), architecture (A), and technology (T), and it introduces a method for analysis of business scenarios based on a clear and structured separation of concerns. The B and O aspects represent the strategic dimension, and the A and T aspects cover the architecture and technology-driven aspects necessary to support the business dimension on the operational level.

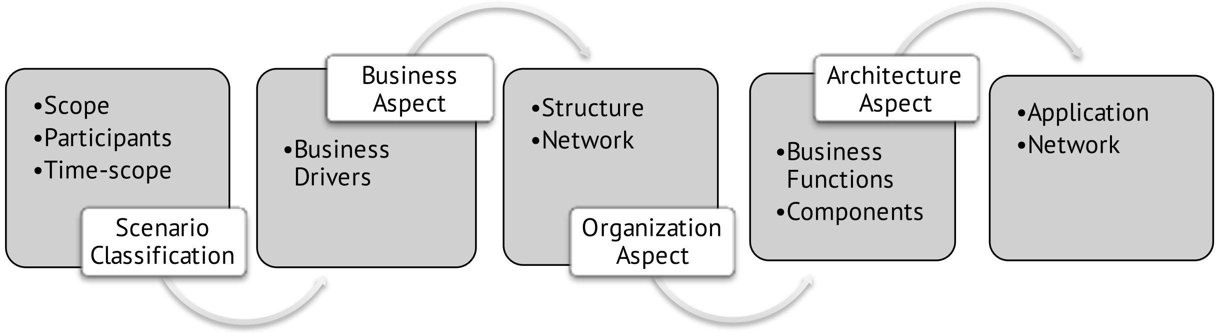


Figure 3.5: Steps of the complex product scenario analysis (based on [Gre15])

As illustrated in Figure 3.5, the analysis begins with classifying complex product scenarios and defining their scope, involved participants, and time-scope as well as features relevant for the analysis of the subsequent aspects of business, organization, architecture, and technology.

Definition 1.3: *Complex product scenarios denote settings in which consumers and providers engage to achieve specific goals represented through the transactions of complex products.*

Accordingly, complex product scenarios represent two-sided interactions between market participants (consumers and providers) and thus are considered business scenarios to facilitate the market exchange of complex products. As indicated in Figure 3.6, the involved participants are linked via complex products (CP_s) they intend to consume (represented by the solid lines) and products/services they intend to provide (dotted lines). Participants might initiate and participate in different scenarios being a consumer or a provider, or both simultaneously. For example, one of the providers involved

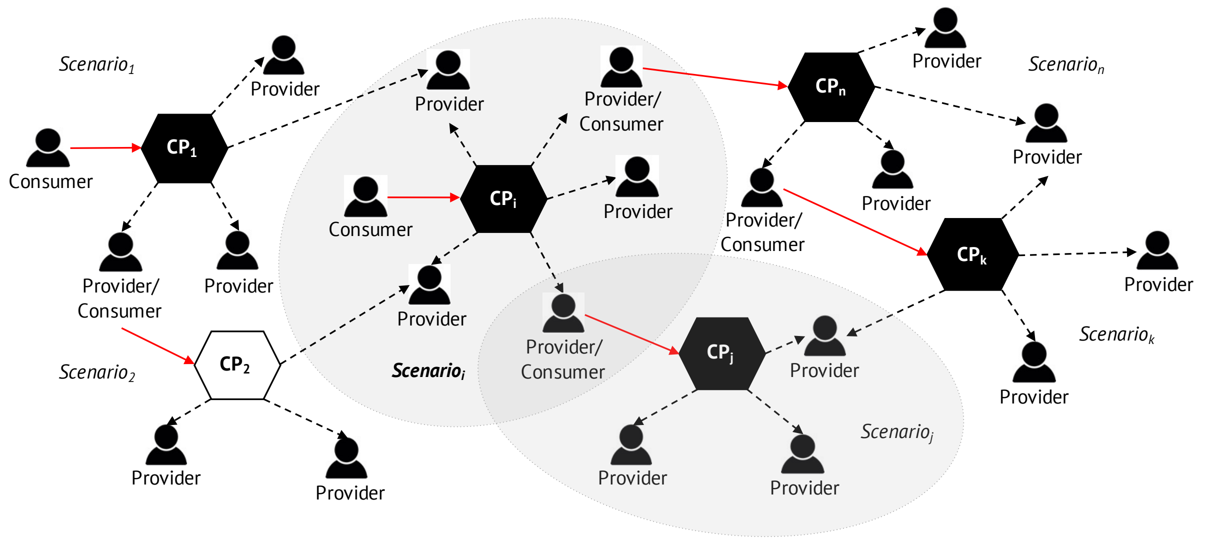


Figure 3.6: Complex product scenarios

in $Scenario_i$ might also start $Scenario_j$ taking the role of a consumer asking for complex product CPj.

As participants in complex product scenarios, consumers and providers can be everybody or everyone connected to the Internet. As illustrated in Figure 3.7 these can be individuals, companies, institutions, as well as software agents and machines acting on behalf of a provider or consumer. Therefore, the scope of complex product scenarios is considered very broad or actor-to-actor, blurring the boundaries of the conventional classifications like business-to-business, business-to-consumer, or consumer-to-consumer scenarios for commercial exchange.

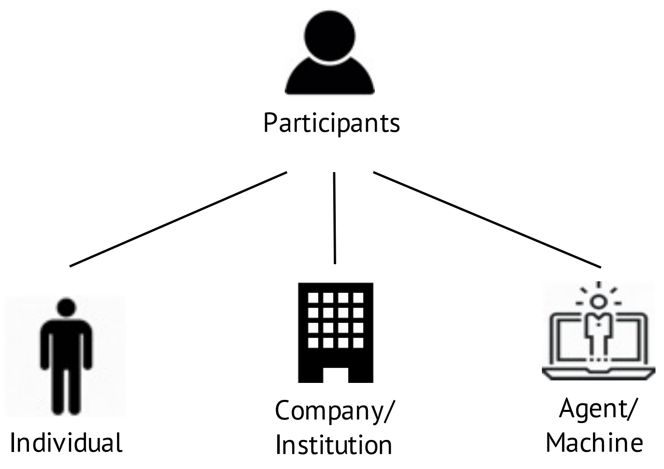


Figure 3.7: Participants in complex product scenarios

The time scope of complex product scenarios is considered dynamic and short-lasting. That is because the interactions between consumers and providers are primarily related to the transaction settlement. As soon as the market transaction of a complex product settles, the relationship between involved participants comes to an end.

Beyond these primary classifications, complex product scenarios are considered *context-centric, distributed*, and *transaction-oriented* business scenarios:

- *context-centric* – as they are initiated by the consumer's demand for a personalized combination of products and services to satisfy a particular need. In the case of our couple, that is a specific set of services (e.g., a babysitter, tickets, a dinner table reservation, and a parking spot) that need to meet constraints such as, e.g., a babysitter from a certified and well-rated agency and a parking spot within walking distance to both locations.
- *distributed* – as consumer's demand might span different business domains (e.g., events, gastronomy, babysitting, parking), complex product scenarios might require many diverse cross-domain providers to deliver distinct parts of the required product/service combination.
- *transaction-oriented* – as its central goal is the commercial transaction of a particular complex product. In contrast to a commercial transaction of an individual product or service, the transaction of a complex product is considered a bundle of single transactions embedded in one enclosing transaction.

Table 3.1 briefly summarizes the above-discussed classification of the complex product scenarios and their main features. It also presents the input for the BOAT analysis and the starting point from which different aspects of complex product scenarios are analyzed.

Table 3.1: Classification of complex product scenarios

Purpose	To support interactions between consumers and providers to achieve a particular goal represented by the transaction of a particular complex product.
Scope	actor-to-actor
Participants	Involved consumers and providers can be everybody or everyone connected to the Internet – individuals, companies, institutions, software agents, and machines acting on behalf of providers or consumers.
Time-scope	Dynamic and short-lasting
Features	context-centric, distributed, transaction-oriented

The following presents the main findings – beginning with the business aspect (Section 3.2.1) and proceeding stepwise to the organization (Section 3.2.2) and architecture (Section 3.2.3) to the technology-related aspects in Section 3.2.4.

3.2.1 Business Aspect

The business aspect of complex product scenarios is determined by the purpose of distributed market spaces to enable market transactions of complex products directly and reliably. Therefore, it considers the perspectives of the participants, consumers, and providers involved, and their main drivers and benefits for participation.

From the consumers' perspective, the previously exemplified complex product scenarios (cf. Section 3.1.2), are already feasible today. However, the settlement process of complex products often causes friction and represents time-consuming tasks consumers need to accomplish manually. Such tasks can get arbitrarily complex reasonably quickly if more auxiliary conditions, constraints, or products and services are added to the desired complex product. Consumers must find their way through a wide variety of offerings while bringing all context-relevant information together and aggregating, comparing, and inferring existing information on their own. This complexity and the current necessity of high consumer involvement lead to adverse selections (cf. Section 1.1) and increases the coordination and transaction costs for complex products. Therefore, the primary drivers of consumers engaging in complex product scenarios are:

- to increase own efficiency in transactions of complex products
- to make them independent in their relationships with the supply-side of the market

Efficiency thereby refers to a cost and time efficiency related to conducting market transactions for complex products. Hence, the first is to obtain efficiencies for complex products comparable to existing utilities enabled by well-established platforms for individual products and services. The second is to regain the control of consumers as initiators of market exchange and make them more independent in their market decisions. For example, market decisions related to the supply-side of the market like choosing potential transaction partners and other parties intermediating between them and providers as the leading actors in commercial exchange.

From the provider's perspective, participants offering products or services need to know where to publish and how to describe offerings to increase the likelihood of being found by consumers looking for such offerings. Since complex products are consumer-driven and usually not predefined, to increase the visibility of their own products and services providers tend to choose well-established online platforms. As previously discussed (cf. Section 1.1), such platforms classically fulfill the role of an intermediary providing a common contact point for aggregated market information. On the one hand, it reduces the transaction costs for providers, but on the other hand it also increases the positional power of such platforms. This puts these platforms in a monopoly position where providers are virtually forced to make use of them and, hence, must accept the rules and terms of these platforms, or they effectively cease to exist in the online marketplace [MJ16]. Furthermore, the understanding of the context in which their own products and services are consumed is considered essential for understanding the value that these products and services contribute for the consumer [Var08]. Such additional information (i.e., knowledge on contextual usage of offered products and services) enables providers to make offerings more attractive to existing or projected consumer segments. Therefore, primary drivers of providers engaging in complex product scenarios are:

- to increase the accessibility of their own offerings
- to increase the level of customer familiarity
- to regain autonomy in their market decisions

The increase of accessibility denotes higher market visibility to various consumer segments looking for such offerings. The level of consumer familiarity refers to a better fit and personalization by providing individual products and services in the consumer-defined context. Autonomy goes in the same direction as stated on the consumer side. It should express more possibilities for providers to connect and contract with consumers directly and without intermediaries as an access point to the demand-side of the market exchange.

3.2.2 Organization Aspect

The organization aspect focuses on the organizational issues, i.e. the organizational structure and business functions necessary to support identified business drivers. To increase efficiency, accessibility, customer familiarity on the one side, and autonomy of decision-making, involved participants (consumers and providers) need to organize as a transaction-oriented structure and follow the peer-to-peer paradigm.

As presented in Chapter 2, the peer-to-peer paradigm denotes an organizational structure of participants (i.e., so-called peers), which are equal in their rights and responsibilities. In the context of an exchange environment, it implies that a transaction-oriented structure builds on top of a peer-to-peer community that enables cooperative interactions among peers with similar interests and rights. Figure 3.8 illustrates such a networked structure, that encompasses three conceptual entities *Peers*, *Demand* and *Supply*.

Peers represent involved participants peers ($Peer_1, \ldots, Peer_n$). Peers can act as a consumer or providers depending on their intention. For example, $Peer_1$ is a consumer asking for a complex product CP_n and the $Peer_n$ is a provider who is offering product P_2. However, the roles are interchangeable and can be taken at the same time, as shown in the case of the $Peer_2$, who is looking for CP_2 and at the same time is offering Service S_2.

Demand represents formulated and published requests for complex products ($CP_1, \ldots, CP_n$), as well as their relationships to published offerings which might be able to satisfy such complex demand. *Supply* represents products ($P_1, \ldots, P_n$) and services ($S_1, \ldots, S_n$). offered by the peers, who are taking the role of providers, as shown on the bottom in Figure 3.8.

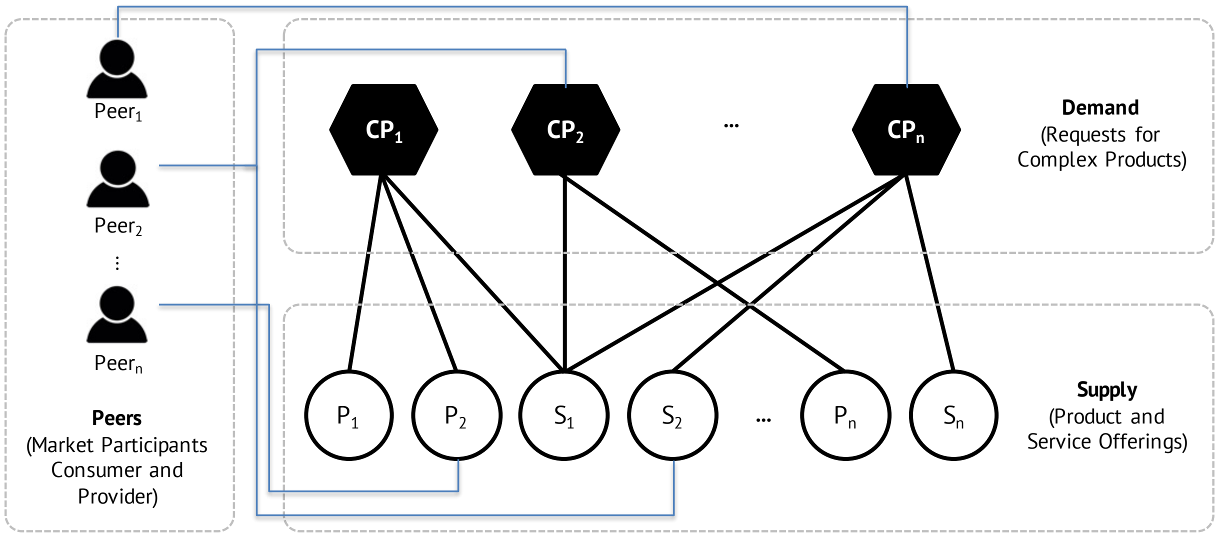

Peer$_1$
Peer$_2$
Peer$_n$
Peers
(Market Participants Consumer and Provider)
CP$_1$
CP$_2$
...
CP$_n$
Demand
(Requests for Complex Products)
P$_1$
P$_2$
S$_1$
S$_2$
...
P$_n$
S$_n$
Supply
(Product and Service Offerings)

Figure 3.8: A transaction-oriented organizational structure facilitated by peers forming a peer-to-peer community

In order to support identified business drivers for its participants (cf. Section 3.2.1), a transaction-oriented peer-to-peer organizational structure must be able to fulfill primary business functions, classified as follows:

- enabling peers to constitute a peer-to-peer community as an exchange environment for transactions of complex products in a direct and trustful manner
- enabling peers to describe and publish offerings, and thus, enabling the constitution of the supply-side
- enabling peers to describe and publish complex product requests, and thus, enabling the constitution of the demand-side
- enabling market transactions of complex products, and thus, facilitating the commercial exchange among peers within a transaction-oriented-oriented peer-to-peer community

Thus a market transaction (see Figure 3.9) represents a process that divides information agreement and settlement [SS98] into phases. As to [PRW03] a market transaction might encompass an additional phase termed as aftersales. Consumers and providers engage in activities such as formulating demand and gathering information about current offerings in the information phase. Matching demand and supply, selecting potential transaction partners, as well as ordering takes place in the agreement phase. Suppose there are possible matches for the demand, and the consumer decides to initiate settlement (i.e., to realize the transaction regarding payment and delivery). In that case, these activities are conducted within the settlement phase. Activities that happen after the settlement of the transaction, such as refunds, reviews, and extended customer relationship management, are conducted as part of the aftersales phase.

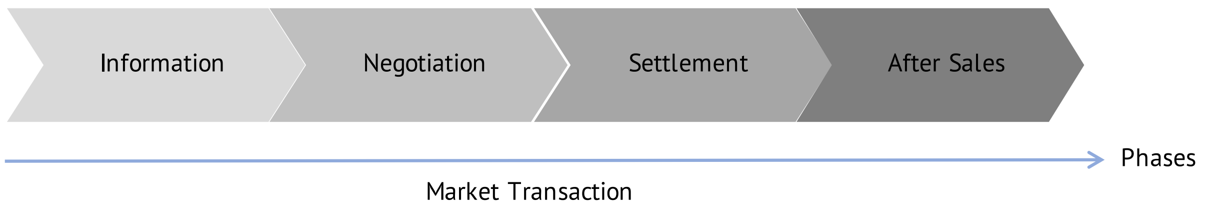


Figure 3.9: A market transaction representing a process that divides into phases information, agreement, settlement and aftersales (based on [SS98],[PRW03])

The identified business functions required to be supported by a transaction-oriented structure are necessary but insufficient. Besides these functions, a transaction-oriented structure built on top of the peer-to-peer community is further needed to implement different mechanisms that inspire trust; since establishing an adequate level of trust and reliance among peers is considered indispensable for peer-to-peer markets [EFL15]. Moreover, such an organizational structure must integrate standards and rules that must be agreed upon and committed among involved peers. Such commonly accepted standards and rules should promote transactions of complex products to achieve a level of reliability considered indispensable for the market exchange in general. Therefore this is to create a satisfactory level of reliability recognized as vital for commercial exchange in peer-to-peer communities, in other word transaction-oriented environments without intermediaries or trusted third parties [XL04].

3.2.3 Architecture Aspect

The architecture aspect focuses on the conceptual structure of an information system expected to support the organizational structure of a complex product scenario on the operational level. It addresses general structuring principles and functional components of a system necessary to support the business functions defined in Section 3.2.2. The focus lies on the market perspective and the conceptual structure of a system at the level where scenario participants, consumers, and providers, engage in market transactions within a peer-to-peer community.

An information system needs to follow the same design principle to support a decentralized organizational model since a system's architecture should primarily relate to its function or purpose. According to this principle known as "form follows function," the requested system has to employ distributed resources in order to perform identified business functions in a decentralized manner. The central architecture concern for such a system considers peers as market participants taking part in the peer-to-peer community and its mechanism, and as the constitutive factors that actively contribute to these mechanisms. As a result, the required system needs to be designed to integrate the peer-to-peer paradigm on the market transaction level and the application and infrastructure levels (cf. Chapter 2 for a detailed explanation of three levels of a peer-to-peer system).

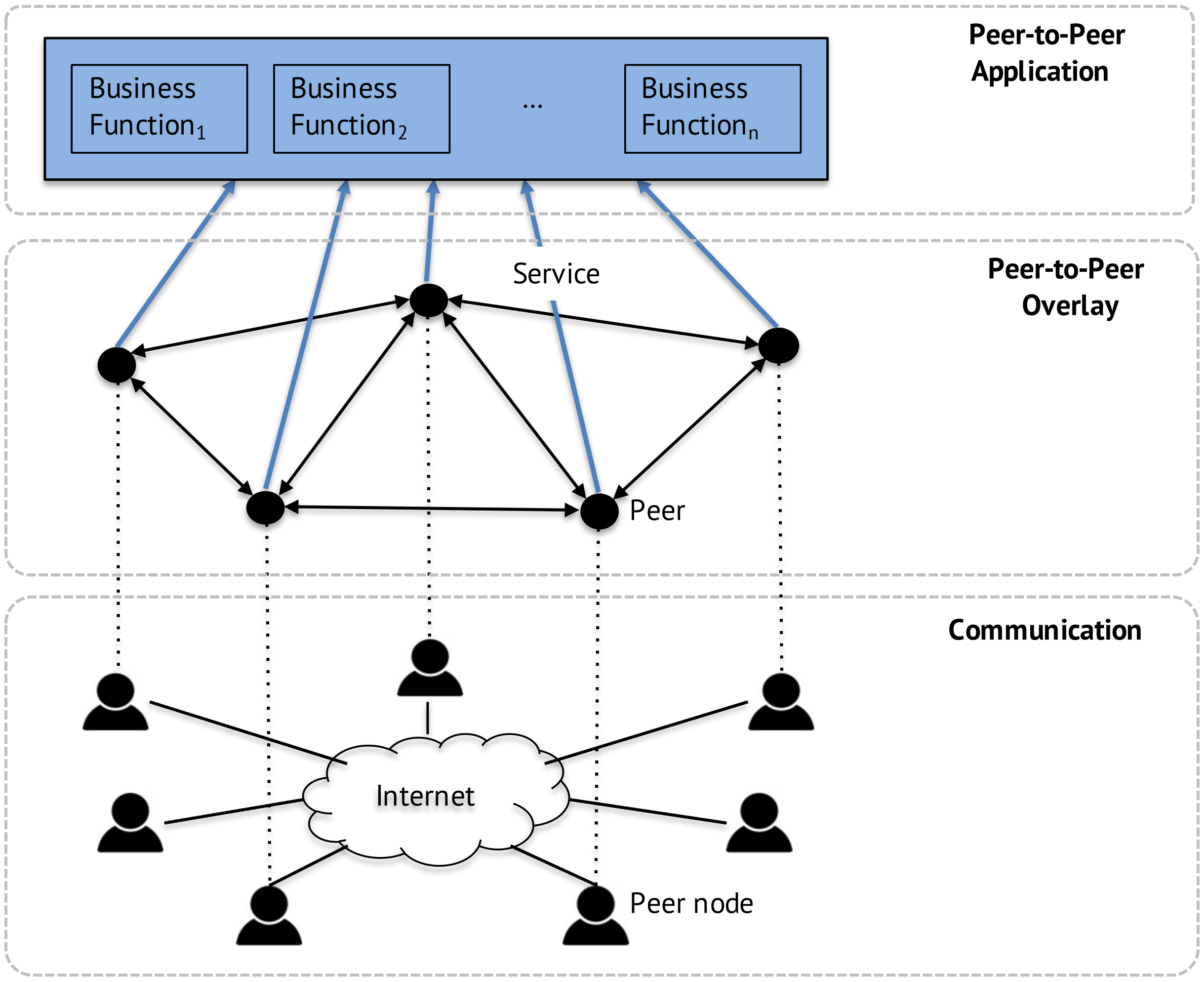


Figure 3.10: Conceptual structure of a system implementing the peer-to-peer paradigm to support specified business functions in a decentralized manner

Figure 3.10 outlines the conceptual structure of a system integrating the concerns and design principles mentioned above. It consists of entities on the application (*Peer-To-Peer Application*) and infrastructure levels represented by a *Peer-To-Peer Overlay*, a virtual network formed by different connections among peers on top of the Internet as the existing communication infrastructure. The application layer represents the utilization of a peer-to-peer overlay in the context of business functions ($BusinessFunction_1$, $\ldots$, $BusinessFunction_n$). It draws on the cooperation of peers in the form of services these peers provide to each other to achieve a common goal, thus facilitating the organization aspect of complex product scenarios. A service (represented by directed blue lines) might be a resource or task provided to support a specific business function. For example, peers might provide resources for storing information, e.g., published descriptions of demanded complex products or offerings on the supply side. Alternatively, peers can provide tasks necessary to support different phases of the market transactions and provide other support services.

Figure 3.11 illustrates an abstract of functional components identified necessary to

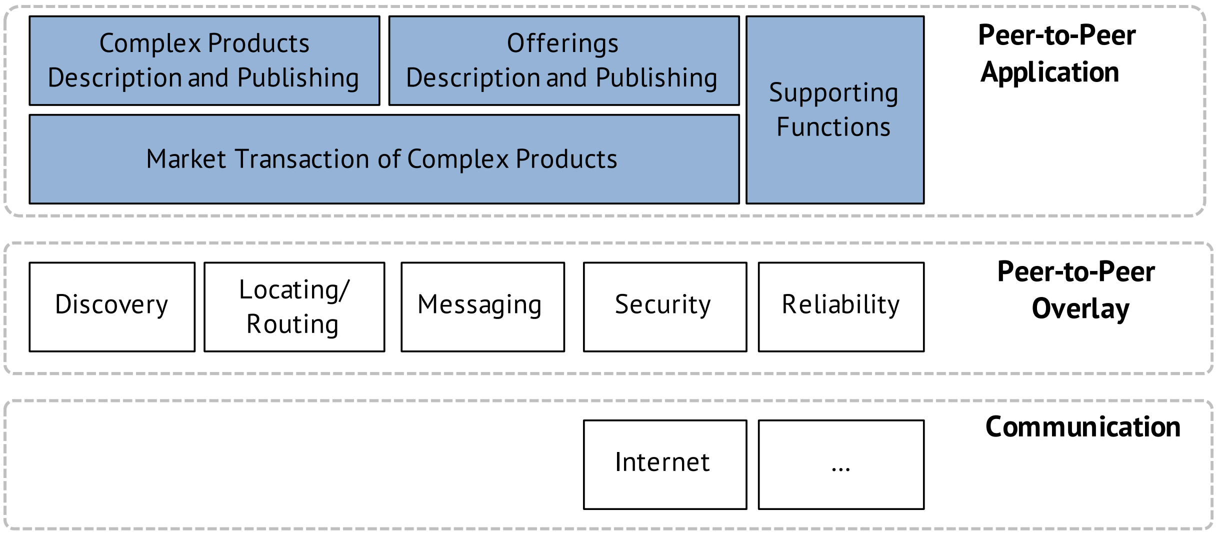


Figure 3.11: An abstract of functional components to support the conceptual structure of a system in Figure 3.10

support the conceptual structure of a system outlined in Figure 3.10.

Communication layer includes components required to cover a broad spectrum of communication paradigms. For example, at one end of the spectrum are computers connected to the Internet, and on the other end are, e.g., devices and sensors connected in an ad-hoc manner via a wireless medium [SGG01].

Peer-To-Peer Overlay layer encompasses components needed for addressing discovery, locating, routing, security, and reliability of a peer-to-peer system (cf. Section 2.2.2). Whereas the discovery component is necessary for the discovery of other peers in a peer-to-peer network, the locating/routing component is needed to optimize the path of a message traveling from one peer to another. While the two components above apply to any peer-to-peer system, the messaging is considered crucial for the collaboration aspect, hence, for sending defined messages among peers and enabling direct interactions across the peer-to-peer community. Security and reliability components are required to address two inherent challenges for the robustness of peer-to-peer infrastructures. Therefore, the primary function of these two components is to ensure that only trusted or authenticated peer nodes should have access to information and services provided by other peers, and guarantee reliable behavior among peer nodes.

Peer-to-Peer Application layer icludes those components critical to concretize the previously stated business functions. That is essential components in describing and publishing complex products (demand-side) and offerings on the supply-side and managing market transactions of complex products. Moreover, additional components, i.e., supportive functions, might be necessary to enable diverse activities related to market exchange within a peer-to-peer community.

3.2.4 Technology Aspect

The technology aspect specifies a set of technology classes considered vital for realizing the system previously defined in Figure 3.10 and thus, for the implementation of functional components summarized in Figure 3.11.

The identified technology classes are shown in Figure 3.12. Internet and Web technologies constitute the communication layer of a peer-to-peer system. Together with peer-to-peer network technology, these two classes provide the basic infrastructure for interaction and collaboration of participants within a market-oriented structure. This market-oriented structure is formed on top of a peer-to-peer network, and represents the ”peer-to-peer community” as previously mentioned in Section 3.2.2.

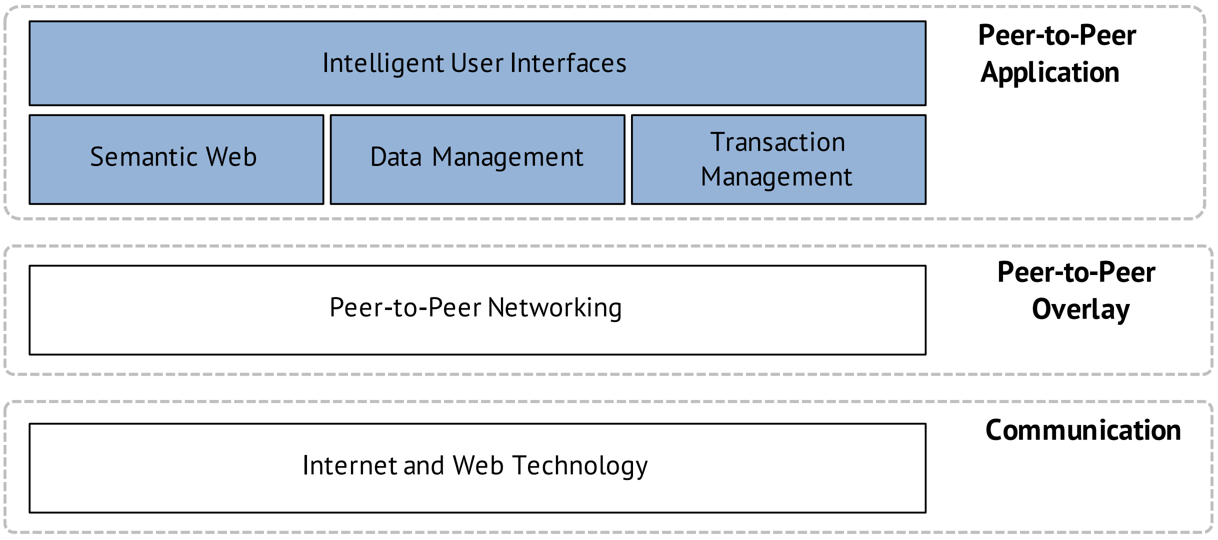


Figure 3.12: An overview of technology classes for the implementation of a system in Figure 3.10

Building on that, Semantic web, data management, and transaction management technology classes are considered necessary for the implementation of functional components of the application layer (cf. Figure 3.11). Semantic Web [Con15] is needed to enable sharing and reusing data in decentralized manner, including using ontologies for the structured description of knowledge in a domain of interest. Moreover, this technology class might support modeling and integrating contextual information of a particular complex product which can be aggregated and derived from interconnected information systems as previously discussed in Section 3.1.2. The data management technology class is required to support specific aspects of organizing, storing, and retrieving information within a peer-to-peer system. Aspects supported by this class of technology are, for example, the interactions between functional components and contents exchanged among these components. Transaction processing management is regarded as necessary for implementing market transactions and activities related to specific phases. To recall, these are negotiation, agreement, settlement, and after-sales activities as indicated in Figure 3.9.

Finally, and as depicted on the top of Figure 3.12, the intelligent User Interfaces (UIs) are seen as essential for the implementation of the system. UIs are the way through which a user interacts with the system and thus the starting point and the initiation of a complex product scenario. Especially, intelligent and flexible UIs are necessary to formulate complex products in a user-friendly and convenient way. To lower the transaction and coordination costs for complex products, intelligent UIs allow data collection of an arbitrary combination of individual products and services that can span over different product/service domains based on the consumer-defined constraints and requirements. Thus the usability of these interfaces plays a key role when it comes to the acceptance of the system as a whole.

Closing remarks on scenario analysis and implications

Table 3.2 summarizes the conducted analysis. It presents the elements identified as most important to each of the analyzed aspects of business, organization, architecture, and technology.

Table 3.2: Summary of a complex product scenario analysis

Aspect	Element
Business	**Purpose:** To facilitate market transactions of complex products directly and reliably. **Drivers:** For a consumer to increase efficiency in transactions of complex products and become independent in relationships with the supply-side of the market. For a provider to increase the accessibility of their own offerings, the level of customer familiarity, and to regain autonomy in market decisions.
Organization	**Networked structure:** To enable interactions among its participants, which are equal in their rights and responsibilities and connected by their intentions to consume or provide complex products.
Architecture	**Design principles and concerns:** Information system deliberately designed to implement specified business functions in a decentralized manner employing distributed resources while supporting trust and reliability. **Functional Components:** Complex Products/Offerings Description and Publishing, Market Transaction of Complex Products, incl. Supportive Functions, Discovery, Locating/Routing, Messaging, Security,Reliability.
Technology	**Supportive Technology Classes:** Internet and Web Technology, Peer-to-Peer Networking, Semantic Web, Data Management, Transaction Management, Intelligent User Interfaces.

The presented elements suggest that the organization and architecture aspects are essential for the nature of the analyzed scenarios. These elements distinguish complex product scenarios from other centrally orchestrated networked scenarios for market exchange and lead to further implications: Firstly, complex product scenarios require a networked (peer-to-peer) organizational structure to enable market transactions for complex products. Thereby, participants are equal in their rights and responsibilities.

They connect through their intentions to consume or provide complex products and thus, pursue the shared purpose realized by market transactions of complex products directly and reliably. Secondly, such a network needs to be supported by the underlying system, which implements specified business functions in order to enable participants to interact and conduct transactions. As a result, the underlying system needs to be designed to employ distributed resources within the network while supporting trust and reliability, considered essential for self-organized networks.

Furthermore, networks of participants organized around a common purpose or value proposition are characterized by its network, shared logic, and a governance system [TA14]. For distributed market spaces, the identified essentials have further implications:

- The value proposition of a market space model is formulated as enabling consumers and providers to constitute and enhance their (distributed) market space as an open exchange environment where they can engage in transactions of complex products directly and reliably.
- The network of a market space model follows the value proposition. This implies that all involved participants need to engage in a cooperative value exchange to realize the value proposition. The value exchange within the network, therefore, needs to be coordinated. Still, the network must remain self-organized as no particular facilitator or "keystone" can control or define the rules of value creation.
- The value creation in a distributed market space rests upon a shared logic. Shared logic designates a shared understanding of the purpose and how it should be realized; hence, it refers to the required business functions and services to support value creation across the value network. For example, all participants need to agree on sharing resources and services to provide the foundation for market exchange without any central coordination and control.
- The governance system in a distributed market space following the principle of decentralized governance. This implies that the supportive functions and coordination of activities, authority structure, and participation control need to be distributed among the underlying network. As a result, participants recognize that they need to collaborate to create value and thus realize the value proposition.

The summarized elements of complex product scenarios and their implications provide insights relevant for the distributed market spaces as an organizing model to support such scenarios. Together with the primary characteristics (cf. Section 3.1) they define the scope of distributed market spaces as a domain and thus impose a set of overall objectives relevant to reference modeling in this domain.

3.3 Overall Objectives

The overall objectives of distributed market spaces as an organizing model to support complex product scenarios are summarized in Table 3.3.

Table 3.3: Overall objectives of distributed market spaces

#	Description
O_1	Enabling market transactions of complex products directly and reliably
O_2	Facilitating decentralization, scalability, and openness
O_3	Supporting simplicity of use and management

Enabling market transactions of complex products directly and reliably (O_1). For consumers, it suggests increasing efficiency, less time and effort in making informed buying decisions, and for providers, gaining higher accessibility for their offers and the level of personalization related to the consumer-defined context.

Facilitating decentralization, scalability, and openness (O_2). To alleviate the adverse effects of the growing platform power and increase the flexibility of integration for a broader range of product/service domains and many involved consumers and providers. Also, promoting peer-to-peer principles and ensuring an adequate level of trust and security is essential for transactions among an open peer-to-peer community.

Supporting simplicity of use and management (O_3). Existing technologies need to be utilized in a way to implement services that are simple to use, easy to integrate, and manage. Thus, a higher level of automation needs to be reached by providing tools and services that support the activities among the participants involved. These can include well-defined, context-aware UIs for the composition of complex products and tools and services for ranking proposals on the consumer side, dynamic service composition, and analytics possibilities on the provider side.

Even though they are general in nature, the formulated objectives (O_1, O_2, O_3) have substantial implications for the reference modeling of distributed market spaces. They lay the ground for defining a frame of reference for the modeling activities and serve as the rationale for deriving the primary design goals on the reference model for distributed market spaces. These will be detailed in Chapter 4. Beyond that, the formulated objectives also serve as the rationale for deriving functional requirements on the operational aspects of an instantiated distributed market space work, which will be presented in Chapter 5.

3.4 Summary

In this chapter, the concept of distributed market spaces is introduced. Therefore chapter 3 presented the first main contribution of this thesis, and thus the outcomes of phase 1 of the modeling process (Figure 3.13).

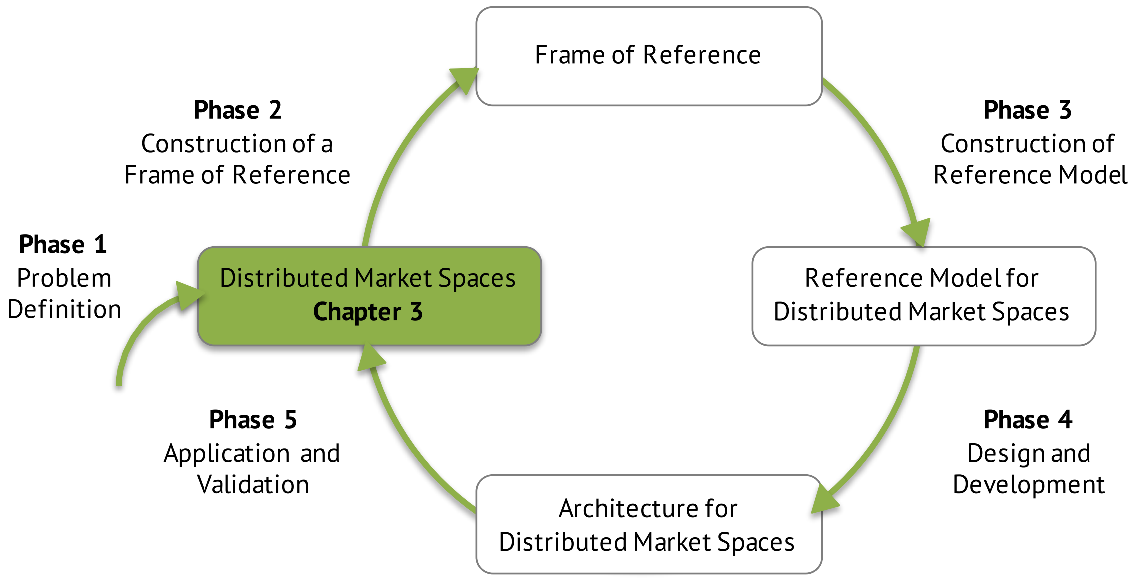

Frame of Reference
Phase 2
Construction of a Frame of Reference
Phase 3
Construction of Reference Model
Phase 1
Problem Definition
Distributed Market Spaces
Chapter 3
Reference Model for Distributed Market Spaces
Phase 5
Application and Validation
Phase 4
Design and Development
Architecture for Distributed Market Spaces

Figure 3.13: Contributions of Chapter 3 to the modeling process

Distributed market spaces are decentralized and self-organized organizing models; their intended use is to encourage market participants (consumers and providers) to constitute and enhance an open exchange environment where they can engage and realize complex product scenarios. The scope of distributed market spaces as the domain of interest for reference modeling is defined by the overall objectives, which derive from analyzing different businesses, organizations, architectures, and technology-related aspects. Overall objectives outline distributed market spaces as exchange environments that enable market transactions of complex products directly and reliably, facilitate decentralization, scalability, and openness, and support simplicity of use and management.

4 | Reference Model for Distributed Market Spaces

This chapter introduces the *reference model for distributed market spaces*. The reference model is meant to serve as a guiding framework for the analysis, design, and implementation of distributed market spaces as self-organized and strictly decentralized online structures for the post-platform economy.

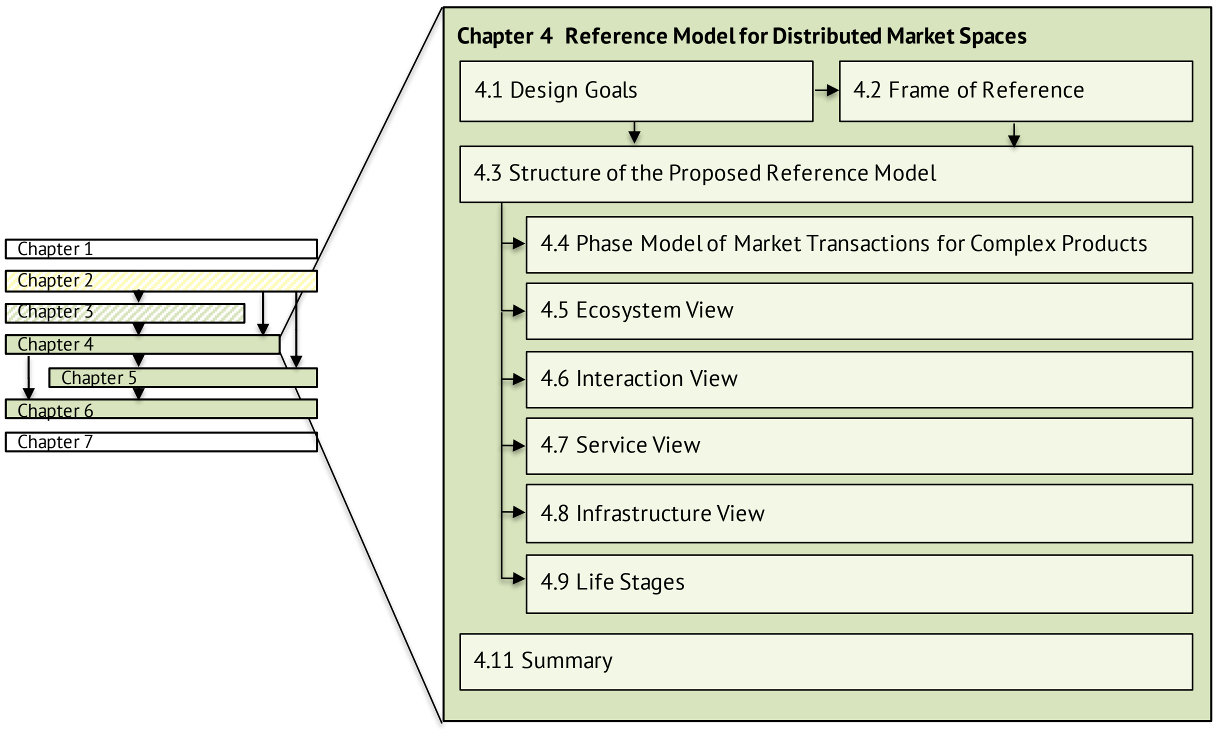


Figure 4.1: Structure of Chapter 4, its content and thematic dependencies

The benefit of the proposed reference model is twofold: First, it informs the understanding of the *distributed market space construct* by defining its core entities, models and elements, and it clarifies the relationships between them. Second, it serves as a set of recommended practices for deriving *distributed market space instances* for the given application and related contextual requirements.

The reference model for distributed market spaces leverages and modifies the St. Gallen Media Reference Model (MRM) [Sch97], [SL98]. It differentiates itself from the reference model by Schmid and Lindemann as it proposes a new dimension, as well as additional models and elements to integrate the specifics of post-platform scenarios. By doing so it addresses the intended purpose and stated objectives of the reference model for distributed market spaces (cf. Section 3). The proposed reference model encompasses three dimensions that define a distributed market space (a reference DMS) in the dimension of *views, phases*, and *stages*. Thus the proposed reference model defines how a reference DMS:

- works on the strategic and operational level,
- enables market transactions for complex products, and
- how its instances might unfold during different life stages.

Figure 4.1 visualizes the structure of this chapter. It begins with Section 4.1 and the description of the primary design goals. Subsequently, Section 4.2 evaluates the related reference models in terms of their applicability, reusability, and adaptability in order to assess to which extent they integrate the required elements, and how these can be reused and adapted for the construction of the reference model for distributed market spaces. Section 4.3 introduces the proposed reference model describing its conceptual structure, the three dimensions, and their core elements. Next, Section 4.4 presents the phase model for market transactions of complex products followed by the description of the ecosystem view in Section 4.5, the interaction view in Section 4.6, the service view in Section 4.7, and the infrastructure view in Section 4.8.2. Finally, Section 4.9.2 introduces the life stages model of distributed market spaces discussing priorities, activities, and challenges related to each of the stages. Chapter 4 closes with a summary in Section 4.10.

The essential parts of this chapter are presented and published in [RWP17], [RP19a], [RP19b] and [RP19c].

4.1 Design Goals

Design goals on modeling the reference model for distributed market spaces (RM_DG_1, RM_DG_2 and RM_DG_3) are presented in Table 4.1. They derive from general requirements on reference modeling and from the special requirements that address the relevant aspects of distributed market spaces as a domain for which the reference model is constructed.

The general requirements for reference modeling apply in principle to every reference model and might include a broader range of quality aspects from constructive fitness, relevance, economics, comparability, systematic construction, language features to the user's perception [Sch13b], [BRS95], [Moo05]. However many of these requirements are rather abstract (e.g., constructive fitness) or subject to the user's perspective (e.g., economics). Still, most of them can be evaluated only through the application in a particular context (e.g., relevance, user perception). Therefore, Ahlemann and Gastl [AG07], and Frank [Fra07] suggest selecting among quality aspects in relation to the perspectives (economic, deployment, engineering, and epistemological perspective), which is considered most relevant for the modeling process. For this work, the engineering and deployment perspective are considered appropriate. The engineering perspective focuses on the conceptual model's feature *technical model features* and *language features*, and the deployment perspective concentrates on the *understandability* as the main factor for the practical usage and thus the success of the reference model. *Technical model features* include the formal correctness of the reference model, architecture, and adaptability, which determines the abstraction level of the reference model [Fra07]. Formal correctness reveals whether the reference model is complete and consistent with the metamodel on which it is based. Hence to what extent it is structurally and behaviorally accurate to the purpose domain. *Language features* refer to the usage of a modeling language. It refers to the level of formalization and extensibility of the used language and the usefulness of conceptual views and tools [Fra07]. *Understandability* reflect the requirement for the reference model to be understandable for the involved users, ideally in a way to correspond to concepts that the prospective users are familiar with [Fra07].

Table 4.1: Design goals of the reference model for distributed market spaces

Rationale	#	**Design Goal**
General requirements on reference modeling	RM_DG_1	Aligning with principles for appropriate reference modeling (*technical model features, language features, and understandability*)
O_1, O_2, O_3	RM_DG_2	Guiding the analysis, design, and implementation of an instance of distributed market space
	RM_DG_3	Assisting in further development of an instantiated distributed market space during its lifecycle

The special requirements, on the other hand, are determined by characteristics of distributed market spaces, which are discussed and detailed in Chapter 3. Special requirements, therefore, address the specific business, organizational, architectural, and technology-related aspects of distributed markets spaces defined by the overall objectives (O_1, O_2, O_3) as stated in Section 3.3.

The formulated reference model design goals (RM_DG_1, RM_DG_2 and RM_DG_3) are used as the starting point to construct a frame of reference for the modeling process, as will be presented in the following section. In addition, the stated design goals serve as the criteria for evaluating the constructed reference model for distributed market spaces, as will be explained in Chapter 6.

4.2 Frame of Reference

The frame of reference for distributed market spaces describes the theoretical background of the proposed reference model. It is comprised of a set of core model elements and chosen construction principles. The set of core elements and necessary building blocks was identified as the result of examining existing reference models regarding their applicability, usability, and reusability for the domain of distributed market spaces. These are presented in Section 4.2.1, followed by Section 4.2.2, describing the chosen construction principles and related language selection, which have been applied within the construction process. Together, both parts represent the used frame of reference that considers the general requirements on reference modeling and the overall objectives of distributed market spaces, and thus the aforementioned design goals (RM_DG_1, RM_DG_2 and RM_DG_3).

4.2.1 Examination of related Reference Models

This section examines existing reference models relevant to the modeling context of distributed market spaces. The modeling context is determined by the prerequisite discussion in Chapter 3 and resulting objectives (cf. Section 3.2). Accordingly, existing reference models for market-oriented environments are reviewed and classified by applying the framework by Braun and Esswein [BE07] to three main classes:

- Practitioner Reference Models
- Scientific Business Process Reference Models
- Scientific Multi-View Reference Models

Besides significant differences between these classes, as will be the discussed in the following, the practitioner models derive from observing many instances available in practice and extracting common elements into reference models, whereas the scientific models leverage and adjust existing reference models.

Each of these three reference model classes has been examined with the focus on their applicability, reusability, and adaptability:

- Applicability – to assess the extent to which they integrate building blocks and model elements required for reference modeling of distributed market spaces.
- Reusability and adaptability – to assess how their elements can be reused and adapted to be of service for the distributed market spaces reference model.

Practitioner Reference Models

Reference models of this class (e.g., [MJ16],[PVC16],[ES16],[RR17], [WLG20]) describe platforms as online structures, which are often implemented in practice and have proven to be very successful in facilitating market exchange over the Internet. Moazed and Johnson [MJ16], Parker, Van Alstyne, and Choudary [PVC16] and Evans and Schmalensee

[ES16] propose models which focus on the anatomy of a platform that consists of four main building blocks: building the audience, matchmaking, providing tools and services, and setting rules and standards. The Rocket Model [RR17] introduced by Reillier and Reillier uses a similar structure adding the data-driven optimizing block considered essential for the strategic development of platforms as ecosystems. Additionally, the model by [RR17] also considers reference modeling in the context of platform life-cycles. It introduces a four-stage model distinguishing between the stages of pre-launch, ignition, scaling-up, and maturity. The PIK model (Platform Innovation Kit [WLG20]) goes in the same direction and offers a comprehensive toolset for extracting platform models for a particular application scenario. PIK proposes nine canvases with predefined questions to support the modeling process. The canvases cover concepts and elements ranging from scanning the environment and describing application scenarios over ideation, defining the value proposition and designing the required platform services to the definition of the strategy for long-term development.

Applicability: The main advantage of practitioner reference models is their recommendation character as they provide solid guidelines to other practitioners looking to conceptualize and instantiate platform models for specific application scenarios. However, neither one of these models makes recommendations about underlying systems and technology-related aspects required to support platforms on the operational level. Concerning their applicability, practitioners reference models fail short to serve as reference models for distributed market spaces as they are single-sided and biased to the perspective of the platform owner. As such they assume centralized ownership structures and miss to include concepts and elements that support aspects of self-organization and self-governance.

Reusability and adaptability: Assessed practitioner reference models provide at least two elements considered useful and adaptable for distributed market spaces. On the one hand, this is a matchmaking service that facilitates market transactions in a way that has proven remarkably beneficial for creating efficiencies, and for building communities and establishing positive network effects ([MJ16], [PVC16],[ES16]). On the other hand, the concept of platform life-cycles by [RR17] can be used and adopted in a way as to distinguish between different perspectives and objectives related to distinct life-stages of distributed market spaces.

Scientific Business Process Reference Models

Reference models summarized in this class (e.g., [BS07],[Fra04],[Kol13],[Aul+16]) focus on the definition of business processes required to support market transactions on the Internet. The H-Model by Becker and Schutte [BS07] introduces a reference model for electronic retail. It describes tasks relevant for the modeling of different parts of a retail enterprise, grouping them in three viewpoints: business functions, processes, and static data. Similar to the H-Model, the E-MEMO reference model Frank introduced by Frank proposes a comprehensive library of process-models for e-commerce. E-MEMO reference model divides process models into different categories based on the business function they cover. These categories range from pre-sales communication, over initiation, pricing, and order processing to customer services related to after-sales. While the aforementioned models [BS07], [Fra04] emphasize inter-organizational processes, Kollmann offers a model that specifies intra-organizational processes of electronic marketplaces (i.e., e-marketplaces). The Kollmann's model defines processes around the phase model of market transactions divided into the phases of information, negotiation, settlement, and aftersales. It further suggests three different views on defined phase-related processes: the view of a consumer, the view of a provider and the view of an intermediary. The intermediary view, is thereby, considered substantial, as an intermediary is considered responsible for providing the underlying infrastructure and integrating processes necessary to facilitate market transactions by means of an e-marketplace.

Applicability: Even though these models provide very detailed guidance on business functions and the related process of an exchange environment, their applicability is limited as they only support the prevailing paradigm of intermediated models with the strictly separated roles of consumers, providers and intermediaries. Moreover, business process reference models fail short to provide any guidelines on how these processes need to be supported by an underlying system. The model by [Aul+16] addresses these issues by introducing a layered E-Commerce Reference Architecture (ERA). ERA proposes processes grouped on three layers: the business, application, and technology layers, as well as processes that define the relationships among business functions across these three layers. Nevertheless, as with other afore-mentioned models, the ERA model does not integrate strategy-related processes showing how an exchange environment needs to be organized and modeled on the strategic level in order to facilitate market exchange.

Reusability and adaptability: Business process reference models provide relevant market transaction processes ([Kol13], [Aul+16]). In particular, phase-related interaction processes can be adapted in a way as to support the specifics of complex products. Especially useful are recommended processes that describe interactions among market participants (consumers and providers) related to return and refund, review and dispute resolution processes.

Scientific Multi-View Reference Models

The class of multi-View reference models summarize conceptual frameworks for the design of market-oriented networked structures that incorporate multiple views (e.g., [Men00], [CA07], [Sch97], [SL98]). The reference model proposed by Menasce [Men00] introduces a four-layer reference model for electronic business. It is composed of a business model, a functional model, a customer behavior model, and an IT resource model. Reference models for collaborative networked organizations (CNO) [CA07]) follow a comparable approach proposing structural, componential, functional and behavioral models, but in contrast to Menasce 's four-layer reference model, the CNO emphasizes the collaboration aspect and integrates models for the design of environments organized around and based on collaborative networks. Although the CNO reference model provides solid foundations on how to conceptualize business environments based on collaborative networks, it misses integrating elements considering the system and infrastructure view, as is the case with the other two reference model classes discussed above.

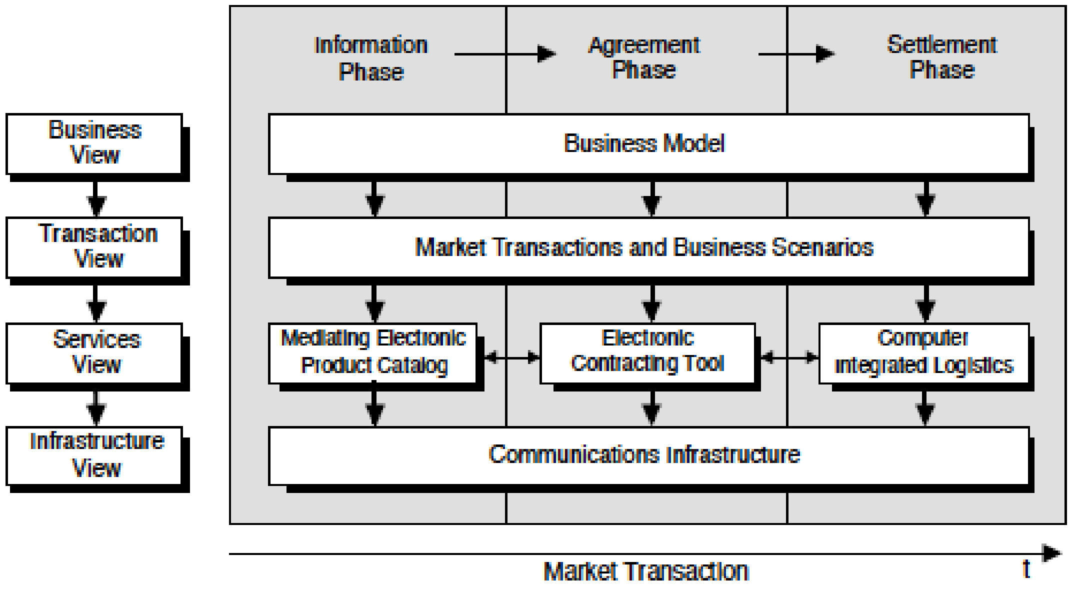

Information Phase
Agreement Phase
Settlement Phase
Business View
Transaction View
Services View
Infrastructure View
Business Model
Market Transactions and Business Scenarios
Mediating Electronic Product Catalog
Electronic Contracting Tool
Computer Integrated Logistics
Communications Infrastructure
Market Transaction
t

Figure 4.2: St. Gallen Media Reference Model (MRM) [Sch97], [SL98]

The St. Gallen Media Reference Model (MRM) [Sch97], [SL98] addresses these shortcomings by introducing a two-dimensional framework for conceptualizing reference media for electronic markets. The term ”media” is related to the concept of a platform as a communication space built for ”social interactions, which allow the participants to meet and which embed them in a common physical, logical, and socio-organizational structure” [Sch97]. The structure of the MRM is presented in Figure 4.2. The horizontal dimension of the MRM represents the market transaction model comprised of the information, agreement, and settlement phases, and the vertical dimension groups the four views. In MRM views are organized into four layers: business, transaction, services, and infrastructure. The business view describes the platform business model with its participants, their roles, and the organizational structure defining the relationships among these roles together with their obligations and rights. The transaction view refers to the relevant transaction-oriented processes and builds upon the underlying service view. The service view comprises all services in the three market transaction phases that need to be available on the platform, and the infrastructure view represents the communications infrastructure required for the implementation of the service view.

Applicability: The MRM has been successfully applied in many different domains (e.g., m-commerce [Sta03], collaborative networks [SS08], service systems [JSS08], enterprise mashup environments [HS09] and marketplaces for cloud services [GPG14]. Regarding its applicability in the context of this work, the MRM is considered suitable to be leveraged as the theoretical framework for the construction of the intended reference model. The rationale behind is twofold: First, it approaches the design process from different viewpoints taking into consideration strategic, operational, service and infrastructure concerns of an exchange environment, and second, it integrates these views with the underlying market transaction model, in order to facilitate market exchange over such environments.

Reusability and adaptability: As a reference for distributed market spaces, the conceptual structure of MRM (both dimensions views and phases) need to be adjusted and modified to cope with the specifics of complex products. Moreover, the MRM reference model only considers the design stage of a market-oriented structure and therefore, needs to be extended with an additional dimension in order to acknowledge different life stages of self-organized and governed online structures.

Closing remarks on related reference models

The above consideration has shown that none of the examined reference model classes meet the requirements of distributed market areas. Instead, each evaluated reference model class contains elements that are deemed useful for reference modeling of the intended reference model. Thus identified elements, which can be summarized as "core elements", outline the frame of reference for constructing the new reference model. As mentioned above, in the case of Schmid and Lindemann’s model, several core elements were identified so that MRM has been selected as a kind of ”initial model” or ”master reference model” from where adjustments and modifications are to be made in order to meet the design goals of the intended reference model for distributed market spaces. Leveraging master models, as recommended by Schütte [Sch13b], aligns to the good reference modeling practices since it might help standardize the wording, language, and core elements, and facilitate embedding new building blocks and elements consistently and clearly.

4.2.2 Applied Construction Principles

Applied construction principles for developing reference models of distributed market spaces follow the reuse-oriented paradigm [Bro07]. As explained in Section 2.1.2, the reuse-oriented paradigm is based on a structured reuse process that follows certain rules. In the case of distributed market areas, this reuse-oriented paradigm is considered appropriate, as it enables a structured modification and expansion of the selected master model and thus lays the basis for the consistency and clarity of the reference model sought.

Although the reuse-oriented paradigm offers several construction principles, and allows for their combination (see Section 2.1.2), instantiation and standardization were considered for application. The principle of initialization was applied to instantiate the core elements of MRM and upon that to either replace or integrate certain model elements to consider specifics of distributed market spaces. For example, to replace an existing view and its models, to add new phases in the phase model or even to add a new dimension to the model. The design principle of specialization was applied to incorporate the general objectives of distributed market spaces as a decentralized and self-organized structure, which were not considered in the initial model. This was the case with revising the business and interaction view, especially with processes that cover core interactions within the ecosystem. By applying these two construction principles, the initial reference model has been modified and extended in a way to meet the design goals of the new reference model. The primary changes relate to the following elements:

- The vertical dimension (*Views*) was modified by replacing the business view with an ecosystem view to in order to integrate the ecosystem perspective of self-organized and governed structures. The service view was modified by adding new services required for implementing the ecosystem view and the associated interaction processes.
- The horizontal dimension (*Phases*) has been extended by rearranging existing and adding a new phase in order to integrate additional activities that are necessary for the realization of market transactions for complex products.
- A new dimension (*Stages*) has been added in order to integrate several aspects belonging to different lifecycle stages of distributed market spaces. The new dimension differentiates between different concerns and activities essential for the development and growth of self-organized and decentralized structures.

Both applied construction principles, instantiation, and specialization are language-independent by design [Vom03],[Bro07] and are implemented accordingly. The modeling activities use the notation of the initial reference model, which is represented by boxes and lines. Moreover, the standardized notation BPMN (BPMN 2.0 [OMG15] common for describing business processes) is used by modeling the functions of the interaction view, as will be shown in the Sections that follow.

4.3 Structure of the Proposed Reference Model

The structure of the reference model for distributed market spaces is presented in Figure 4.3. It encompasses three dimensions defining a reference distributed market space (*a reference DMS*) in the dimension of *Views, Phases*, and *Stages*. Together they build *a comprehensive multi-dimensional and multi-view reference model*, that describes how a reference DMS:

- works on the strategic and operational level
- enables market transactions for complex products, and
- how its instances might unfold during different life stages

Views dimension describes a reference DMS taking four different points of view:

- Ecosystem view maps the organizational structure of a reference DMS as an ecosystem. It proposes an ecosystem model that outlines actors, their roles, and primary activity flows and explicates how identified actors and activities need to link and align in order for the ecosystem's value proposition to materialize.

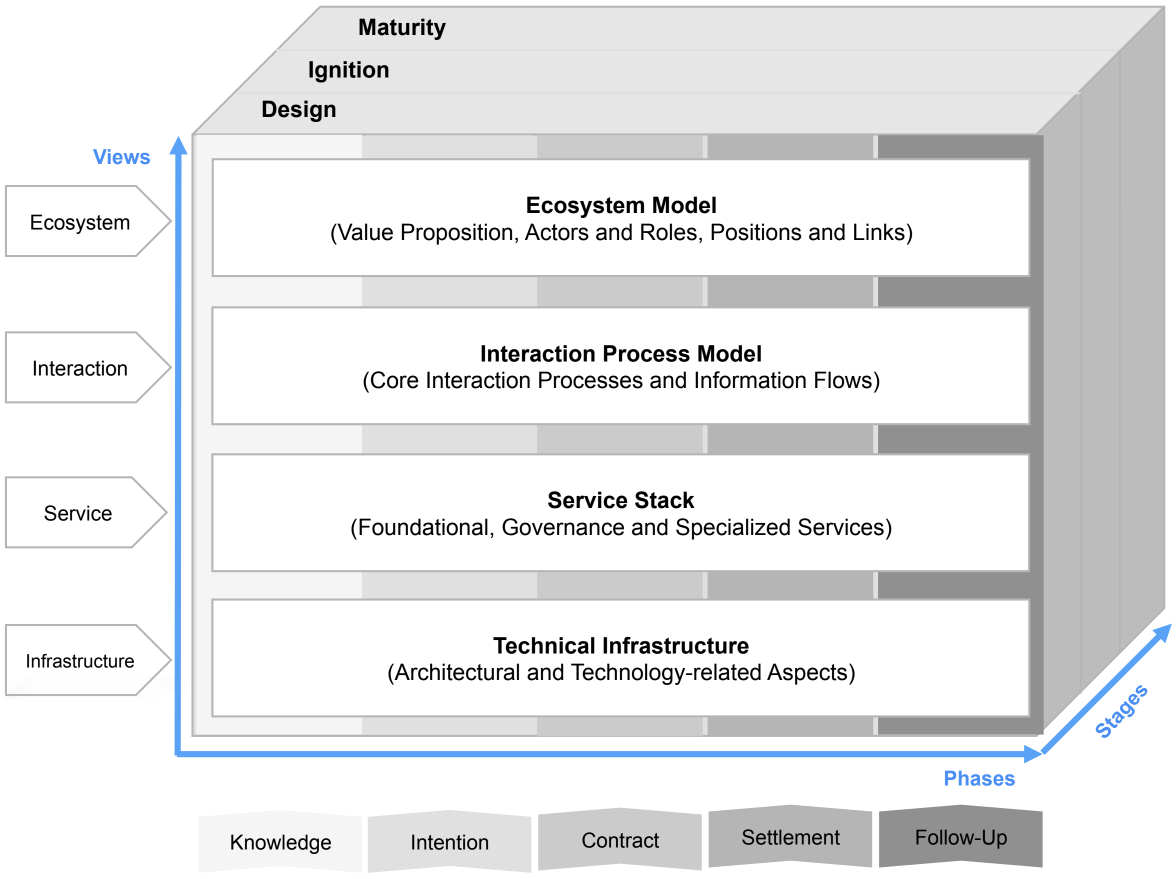

Maturity
Ignition
Design
Views
Ecosystem
Interaction
Service
Infrastructure
Ecosystem Model
(Value Proposition, Actors and Roles, Positions and Links)
Interaction Process Model
(Core Interaction Processes and Information Flows)
Service Stack
(Foundational, Governance and Specialized Services)
Technical Infrastructure
(Architectural and Technology-related Aspects)
Stages
Phases
Knowledge
Intention
Contract
Settlement
Follow-Up

Figure 4.3: Reference model for distributed market spaces

- Interaction view specifies the core interactions among identified actors taking different roles at the operational level of a reference DMS. It proposes an interaction process model that specifies the interaction processes, relevant activities, and resulting information flows required for market transactions of complex products through a reference DMS.
- Service view defines services that a reference DMS must provide to its participants in order to facilitate the ecosystem and interaction views. It introduces a service stack that implements the ecosystem model and its core interactions processes specified by the interaction phase model.
- Infrastructure view describes the technical infrastructure of a reference DMS for the implementation of the service view. It considers architectural and technology-related aspects and comprises the groundwork for the implementation of the defined service stack.

Section 4.5 provides a detailed description of the ecosystem view and the proposed ecosystem model, followed by the description of the interaction, service and infrastructure view in Section 4.6, Section 4.7, and Section 4.8.2.

Phases dimension defines a reference DMS as a market-oriented environment for supporting transactions of complex products. As shown in Figure 4.3 it is based on a phase model for market transactions of complex products, and it comprises the knowledge, intention, contract, settlement, and follow-up phase. The purpose of the phase model is to define how to initiate, arrange and settle contractual agreements for market transactions of complex products in the most efficient manner. Whereby, efficient manner refers to lowering transaction costs for consumers looking for transactions of complex products directly and reliably. Section 4.4 introduces the phase model for market transactions of complex products and describes each of the phases including their outcomes.

Stages dimension comprises life stages a reference DMS might undergo during its development and growth. As shown in Figure 1.1 these are the design, ignition, and maturity stages. Each of them focuses on different concerns and therefore has different priorities. While the main priority of the design stage is to blueprint, prototype and launch a DMS for a specific application domain, the priority of ignition stage is to build the critical mass of participants as the prerequisite for the maturity stage. If the maturity stage is achieved, the main priority could be to retain existing participants (network) and connect to others for sustainability and growth. Section 4.9.2 specifies these three stages by describing their concerns and priorities, suggesting activities related to each of them.

4.4 Phase Model of Market Transactions for Complex Products

The phase model of market transactions for complex products defines the necessary interactions between market participants, consumers, and providers, engaged in transactions of complex products. Hence, it lays the ground for lowering transaction costs for consumers looking for transactions of products over DMS.

The proposed phase model enhances the existing market transaction model as used in MRM [Sch97], [SL98]. The extensions refer to:

- The integration of the ’Follow-Up’ phase, an additional phase which integrates interaction processes among market participants that happen after the settlement.
- The integration of additional processes which addresses the specifics of complex products (cf. Section 3.2)

As a result, the phase model of market transactions for complex products encompasses five phases: *Knowledge, Intention, Contract, Settlement*, and *Follow-Up*.

Each of these phases represents a group of activities by involved participants, and each of them has a defined output or a phase result. Figure 4.4 presents the proposed phase model and summarizes the results of each of the five phases as described in the following.

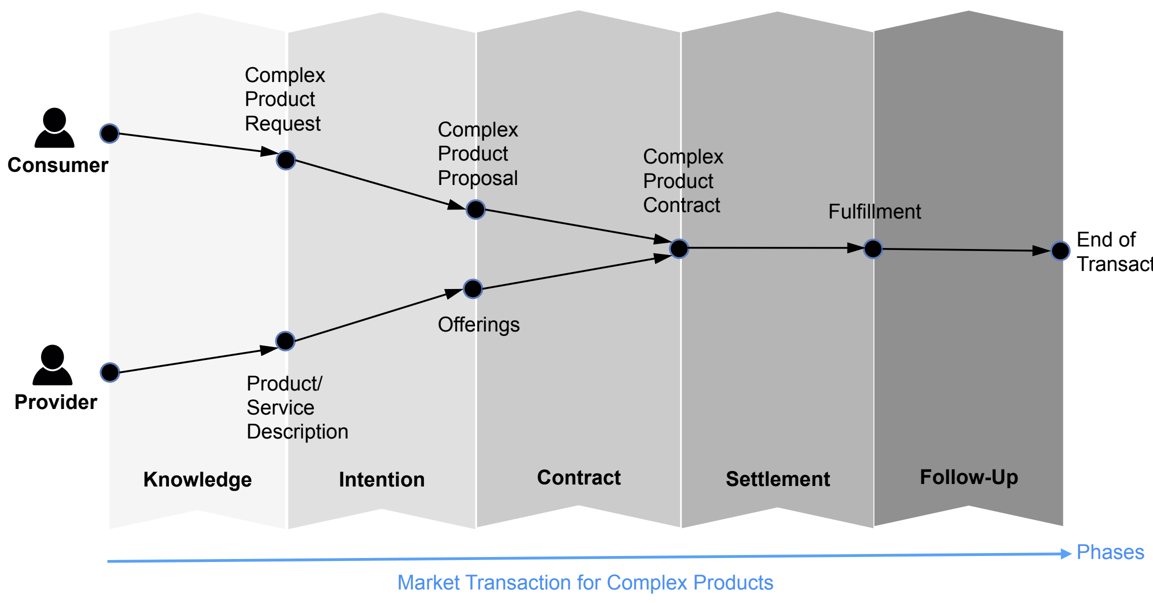


Figure 4.4: Phase model of a market transaction for complex products

- In the *Knowledge phase,* market participants acquire an overview of the supply and demand within a distributed market space. Providers publish their offers by publishing descriptions of products and services they offer. Consumers formulate their demands as complex product requests, and, based on them, search for potential providers that can provide parts of the required complex product. Consequently, the knowledge phase ends with a product/service description and a formulated complex product request accompanied by a list of possible transaction partners.
- In the *Intention phase,* market participants negotiate conditions for an agreement for the particular complex product. It covers the process of sending "Requests for Offer" (RfO) to the potential transaction partners (consumer), and providers send back specific offerings, including price tag, payment mode, and delivery conditions. Then consumers aggregate all offerings received, and create complex product proposals, including ranking them based on the defined requirements and constraints. Consumers might then select one complex product proposal, which best suits their demands. The chosen complex product proposal (consumer side) and offerings (provider side) represent the phase results and are the starting point for the next phase, forming the basis for the contractual agreement to be made.
- In the *Contract phase,* consumers and providers concretize the negotiated agreement represented by a legally binding contract. From the consumer side, a complex product contract is considered as an umbrella contract since it incorporates different arrangements for different parts of the complex product. The umbrella contract represents a one-to-many contract situation and requires consumer's involvement in several contractual processes (one for each product or service).

On the provider side, the contracting process is considered as a one-to-one contract situation with an additional activity regarding the confirmation of a pending contract. The phase ends with an agreed legally binding contract for a complex product, which is the starting point for the settlement phase.

- The *Settlement phase* serves to fulfill the obligations resulting from the complex product contract agreed in the contract phase. Similar to individual products and services, the settlement phase of complex products encompasses interaction processes related to delivery, payment, and logistics. Depending on the type of exchanged product or service as well as the involved providers, the settlement phase might include additional sub-processes related to the type of settlement such as trusted third-party or trustless settlement (as detailed in Section 4.6.4.)
- The *Follow-Up phase* closes the market transaction for complex products. As the fifth phase, follow-up supports interactions between transaction partners that happen after settlement. These are interactions and activities related to reviews of settled transactions, customer support, management of return and refund, as well as management of disputes among transaction partners.

The proposed phase model of market transactions for complex products is considered in more depth in Section 4.6. Therein, inner workings of each phase, their processes, and related activities are detailed and presented in the context of the interaction view of a reference DMS.

4.5 Ecosystem View

The ecosystem view maps the ecosystem structure of a reference DMS. *Ecosystem Model* blueprints the proposed ecosystem structure by outlining the primary activities, actors and their roles, and how actors and activities need to link and align in order to support the shared purpose of the DMS ecosystem.

Ecosystems refer to "economic communities supported by a foundation of interacting organizations and individuals [Moo93], that use common standards and collectively provides goods and services [Tee16]". As such, ecosystems consist of a large number of loosely interconnected actors who create value through the process of cooperation and competition [IL04b],[Tia+08]. This explicit dependence of involved actors who rely on each other is considered the essential feature of ecosystems that distinguishes them from other interconnected environments, e.g., value networks or value chains.

As organizational models, ecosystems are defined by two primary characteristics: First, how value is created, and second, how it is shared, in order to satisfy the individual and collective motivation of actors participating in the ecosystem [IL04b], [Ron+15], [Tia+08], [EP13]. Consequently, ecosystem models are considered constructs composed of entities and elements required to specify how value is created and shared among participating actors.

Literature provides various approaches and concepts in order to formalize ecosystem models such as e.g., BEAM [Tia+08], MOBENA [Bat+13], 6c [Ron+15], VISOR [EP13], Value Design [WLR14], Ecosystem Construct [Adn17]. Even though each of these modeling approaches has its initial focus, in general, they are addressing ecosystem modeling from two different perspectives [Adn17]:

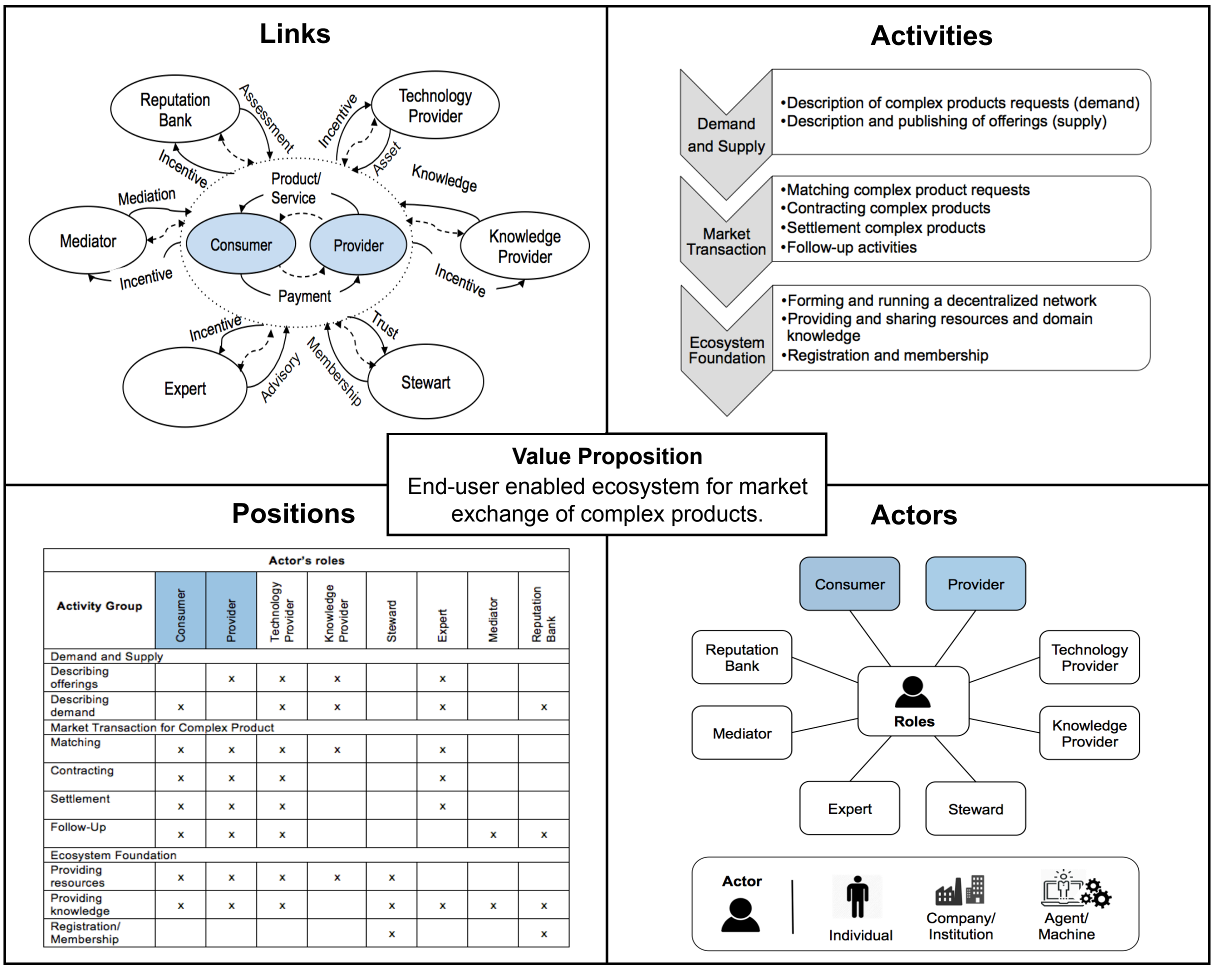
- Ecosystem-as-affiliation – viewing ecosystems as communities of associated actors defined by their network affiliation and are a complement to a focal actor [Tia+08],[Bat+13],[Ron+15],[EP13].
- Ecosystem-as-structure – viewing ecosystems as alignment structures of activities and actors defined by a shared value proposition, rather than being an affiliate to a focal actor [Adn17],[WLR14].

For the modeling of an ecosystem for reference DMS, the ecosystem-as-structure perspective applies, and consequently, the modeling approach by [Adn17] concerns. The rationale behind lays in the definition and requirements of a reference DMS. As a self-organized and governed structure of actors with equal rights and responsibilities, the DMS ecosystem needs to be organized in a way to enable the shared value proposition, which is to be realized in a decentralized manner. Since there is no focal actor, this requires that actors align following an agreement on how value is created and shared within the ecosystem they constitute. Such an alignment, thus, refers not only to shared motivation and incentives as is the case with ecosystems-as-affiliation, but also requires the actors' consistent engagement. As will be discussed below, a consistent engagement entails a commitment to taking different roles, and by providing resources and services to uphold the ecosystem.

Figure 4.5 presents the resulting *Ecosystem Model* for reference DMS. *Value proposition* defines the shared purpose of the DMS ecosystem and is formulated as an *end-user enabled ecosystem for the market exchange of complex products directly and reliably*. Core elements that underlie the stated value proposition are:

- *Activities* define the primary activity groups and discrete actions to be undertaken
- *Actors* specify the entities that undertake these activities taking different roles
- *Positions* specify where in the flow of activities actors are located
- *Links* specify how actors taking different roles need to interact and what value they need to exchange

Core elements mutually depend on each other and together describe how value is expected to be created and shared within the ecosystem. Hence, they blueprint a decentralized environment of interdependent collaboration that is the organizational structure of the DMS ecosystem underlying the stated value proposition.



Actor's roles								
Activity Group	Consumer	Provider	Technology Provider	Knowledge Provider	Steward	Expert	Mediator	Reputation Bank
Demand and Supply								
Describing offerings		x	x	x		x		
Describing demand	x		x	x		x		x
Market Transaction for Complex Product								
Matching	x	x	x	x		x		
Contracting	x	x	x			x		
Settlement	x	x	x			x		
Follow-Up	x	x	x				x	x
Ecosystem Foundation								
Providing resources	x	x	x	x	x			
Providing knowledge	x	x	x		x	x	x	x
Registration/ Membership					x			x

Figure 4.5: Ecosystem model of a reference DMS

4.5.1 Activities

There are three primary activity groups that the DMS ecosystem needs to support to realize the stated value proposition:

- Supply and Demand
- Market Transactions
- Ecosystem Foundation

The *Supply and Demand* activity group defines activities related to the composition and description of complex product requests on the consumer side (i.e., demand), and the description and publishing of products and services on the provider side (i.e., supply).

The *Market Transaction* activity group defines activities to support the phase model of market transactions for complex products. As previously described (cf. Section 1.3), these are activities necessary to support interaction processes in each of the phases of negotiation, contracting, settlement and follow-up complex products.

The *Ecosystem Foundation* activity group defines activities to build the foundation that are essential for setting up and operating the ecosystem. It includes forming and running the network by providing resources and services (e.g., hosting, tools) and domain knowledge necessary for market exchange in a specific domain (e.g., domain ontologies and vocabularies for that particular domain).

4.5.2 Actors and Roles

Actors in DMS ecosystem can be anyone or anything connected to the Internet intending to engage in complex product scenarios. That includes individuals, companies, institutions or associations, and other networks, as well as autonomous actors such as software agents or machines (cf. Section 2.2). As shown in Figure 4.6, there are (at least) eight roles that actors can take:

- Consumer
- Provider
- Technology Provider
- Knowledge Provider
- Steward
- Expert
- Mediator
- Reputation Bank

Consumer and *Provider* are considered shaper roles as they shape the value proposition and thus the birth of the DMS ecosystem. Other roles mentioned (*Technology Provider, Knowledge Provider, Steward, Expert, Mediator*, and *Reputation Bank*) are enabler roles.

Their purpose is to enable the ecosystem to provide comprehensive services to the shaper roles. Therefore the primary function of enabler roles is to enable the ecosystem's value creation by undertaking activities as defined in previous Section 4.5.1.

Roles are considered to be motivated by two factors. On the one hand, they are motivated by a shared purpose in order to realize the ecosystems' value proposition, and on the other hand, they are driven by individual motivation. Shared motivation entails the commitment to be a constitutive part of the ecosystem's alignment structure and to continuously engage in order for the ecosystem to uphold. The individual motivation indicates the additional value an actor expects from the participation in the DMS ecosystem. Such an expected value might differ from role to role, be subject to various actor types and even change over different life stages of the DMS. Even though each of the roles has a different function and is responsible for different activities, roles can overlap and be taken concurrently, as they do not exclude each other. For example, a consumer (shaper role) can also take the role of, e.g., technology provider, or an expert (enabler role) at the same time.

Table 4.2: Actor's roles and their motivation, or expected value from their involvement within the ecosystem

Role	**Description**	**Motivation/Expected Value**
Consumer	Looking for a complex product.	To satisfy personalized needs defined through market transactions of complex products directly and reliably.
Provider	Offering products or services in one particular domain or many domains.	To earn revenue per product/service sold (payment). To increase visibility of offerings. To increase the level of customization based on contextual information provided by consumer.
Technology Provider	Providing technology assets (resource/-tool/service) to support market transactions of complex products.	To contribute to the ecosystem foundation. To earn revenue by guaranteeing availability only to paying users (incentive). To leverage usage-data for improvement and developing new assets.
Knowledge Provider	Providing domain knowledge.	To contribute to the shared knowledge base. To earn revenue by providing paid knowledge-based services (incentive).
Steward	Registering ecosystem's members after being granted access.	To contribute to self-governance capability of the ecosystem. To ensure congruence between members, rules and norms.
Expert	Offering expertise and advice to inform decision making.	To earn revenue through advisory and user's feedback (incentive).
Mediator	Offering mediation to support resolving disputes and conflicts.	To earn revenue through mediation and user's feedback (incentive).
Reputation Bank	Assessing ecosystem's members regarding their reliability, solvency, and worthiness.	To capture two-sided reviews about conducted transactions needed for qualified assessment of members (assessments). To promote an adequate level of trust among ecosystem's members.

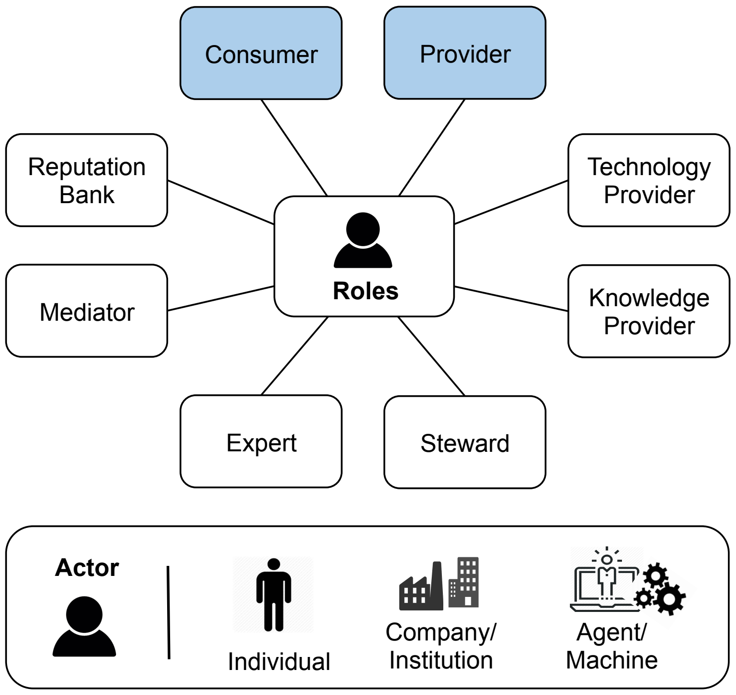


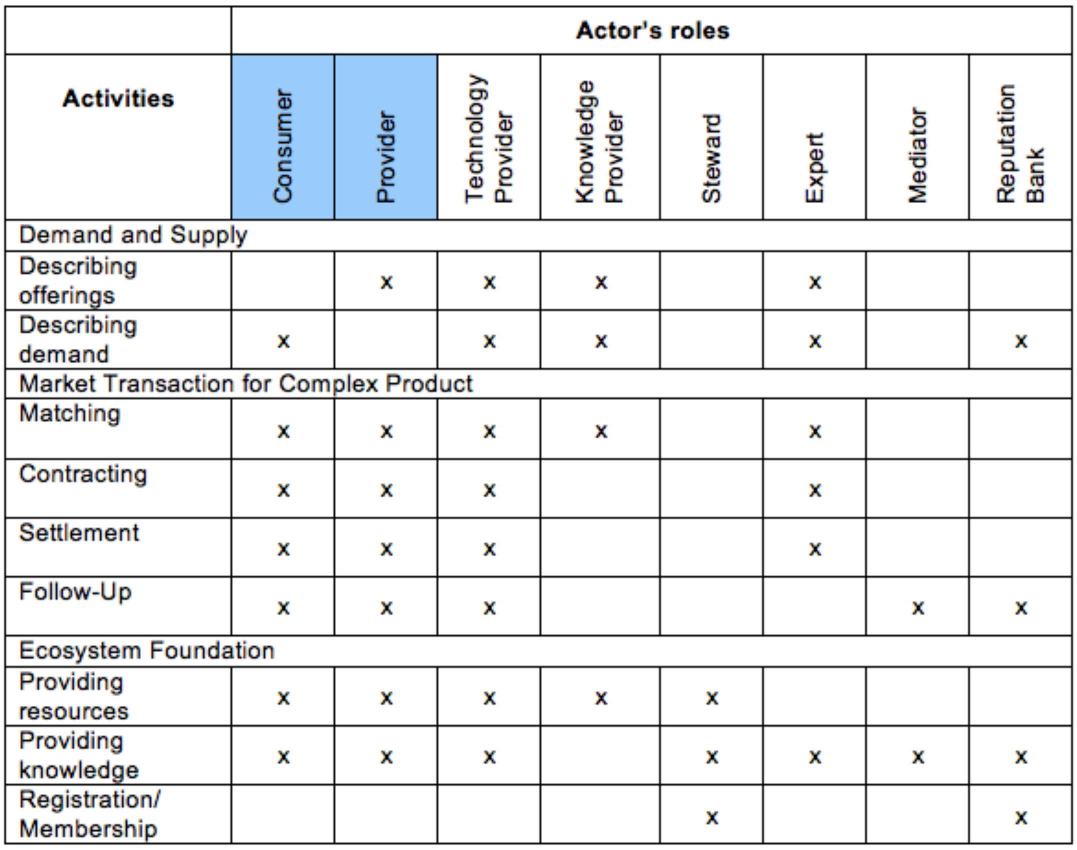
Figure 4.6: Actors and their roles

Table 4.2 summarizes the identified roles describing their functions and stipulating possible individual motivation or expected value from the participation in the DMS ecosystem.

4.5.3 Positions

For the stated value proposition to realize, the definition of the necessary activities and identifying actors who need to undertake these activities as well as naming their roles are necessary, but not sufficient. To create value, ecosystem's actors taking different roles need to align around the activities and take a particular position in the overall value creation.

Positions, as illustrated in Figure 4.7, provide an overview of where in the flow of activities identified actors need to locate. Single roles can contribute to several activities, and specific activities might require several roles to engage. For example, to support consumers in formulating demand, that is, composing complex products as arbitrary combinations of individual products and services, several roles need to engage. Besides the consumer who initiates the process, the technology and knowledge provider are required to provide tools and knowledge in order to enable the composition of complex products and the integration of contextual information. Depending on the complexity and level of personalization, the composition of the complex product might also involve further roles. In the example, experts might support composing the required product/service combination, and reputation bank might provide information about



Activities	Consumer	Provider	Technology Provider	Knowledge Provider	Steward	Expert	Mediator	Reputation Bank
Demand and Supply								
Describing offerings		x	x	x		x		
Describing demand	x		x	x		x		x
Market Transaction for Complex Product								
Matching	x	x	x	x		x		
Contracting	x	x	x			x		
Settlement	x	x	x			x		
Follow-Up	x	x	x				x	x
Ecosystem Foundation								
Providing resources	x	x	x	x	x			
Providing knowledge	x	x	x		x	x	x	x
Registration/ Membership					x			x

Figure 4.7: Alignment structure of actors taking different roles and activities

reputation and worthiness of the possible providers. In that way enabler roles are supporting consumers proceeding in a more targeted manner, narrowing the selection of the potential providers already at the beginning of the market transaction, and thus, contribute to lowering of transaction costs.

4.5.4 Links

Links illustrate how actors taking different roles need to interact and specifies the transfer between them. Figure 4.8 visualizes links in the form of a flow diagram, which outlines the overall pattern of exchanges within the DMS ecosystem. The focus lies on the roles of shaper, consumer and provider, and the visualization of the most important interactions with enabler roles and resulting exchanges.

The nodes represent actors performing a particular role, and the arrows the essential interactions indicating the ”value exchanged” between these roles. Solid lines denote ”tangible” exchanges, such as product/service delivery or payment, as is the case with the consumer and provider roles (see Figure 4.8). Dotted lines indicate additional value exchanged considered ”intangible”, for example, feedback, reputation or usage-related

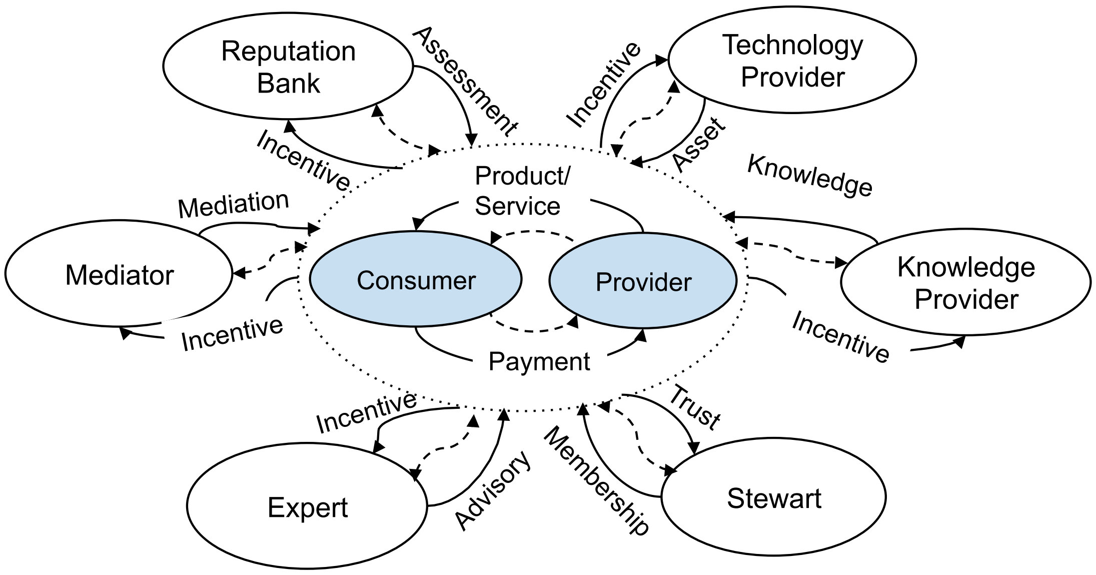


Figure 4.8: Links – showing interactions between shaper and enabler roles and resulting value exchanged

data. Regarding the interaction between shaper roles, this might be the contextual information a provider might receive from a consumer requesting an individual product or service. Such additional information is considered valuable as it can be used to increase the contextualization of offerings, and thus, enables the provider to provide in the consumer context. Vice-versa, a consumer can get personalized bundles of product or service that best fits his demand based on the provided contextual information.

Links explicate the overall value exchange within the DMS ecosystem necessary to realize its proposition and thus the shared motivation of named roles. Further, links consider the content of transfers required to satisfy the individual motivation of role, i.e., the expected value from the participation in the DMS ecosystem (cf. Table 4.2). Note that Figure 4.8 presents only the most important tangible exchanges. For an in-depth discussion see Appendix I.

4.6 Interaction View

The interaction view specifies the core interactions required for market transactions of complex products through a reference DMS. The purpose of the resulting *Interaction Process Model* is to define critical processes between shaper roles (consumers and providers) structured around the phase model of market transactions for complex products.

Figure 4.9 shows a high-level overview of the proposed process model. It presents the core interaction processes between a consumer looking for a complex product, and a

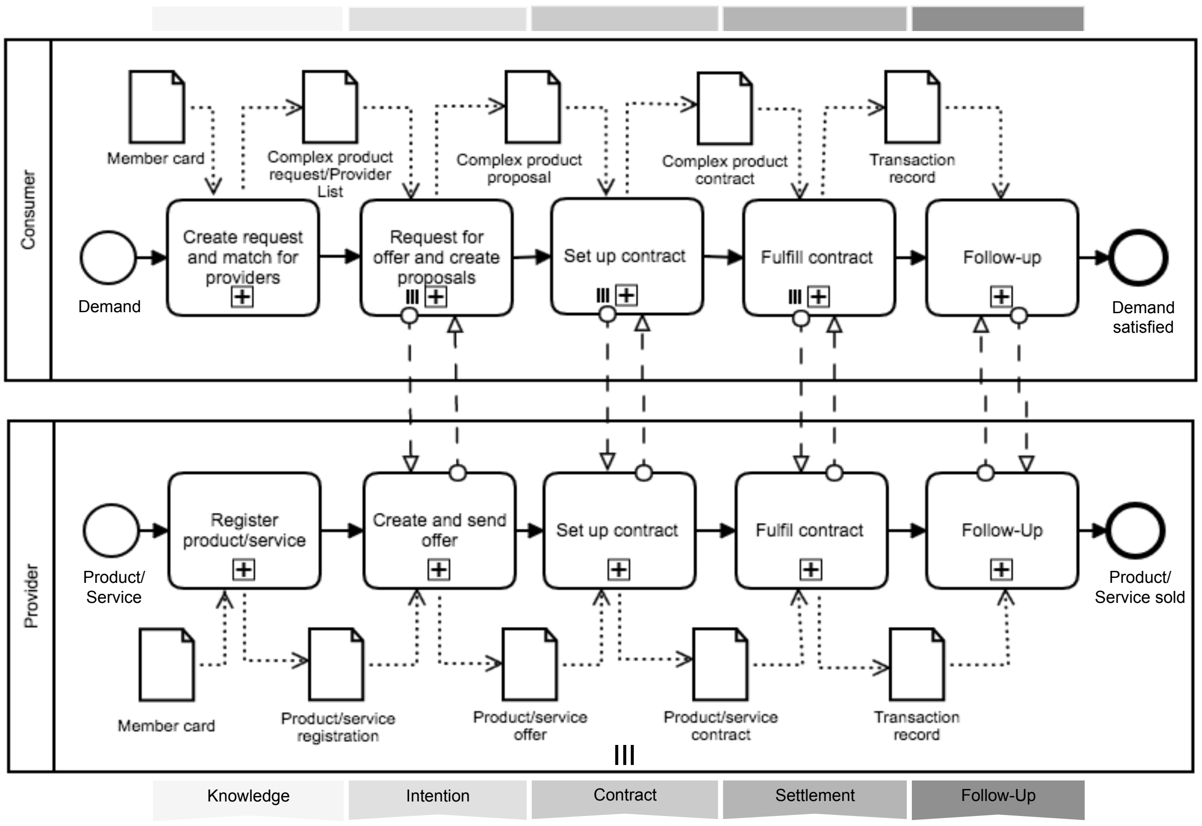


Figure 4.9: Interaction Process Model of a reference DMS - a high-level overview as BPMN2.0 diagram

provider (or many of them) engaged in the market transaction for that particular complex product. It summarizes the necessary processes for each phase and the resulting information objects, explicating their relationships and locations inside each of the phases of the market transaction model (cf. Figure 4.4). The high-level overview and all related sub-processes are modeled and described using the Business Process Model Notation (BPMN 2.0 [OMG15]).

As indicated in Figure 4.9, the high-level interaction process starts with the demand for a particular complex product on the consumer side and the idea for a concrete product or service on the provider side. The prerequisite for the participation in the process is DMS membership. DMS memberships are represented by *Member cards*, which are the basis for all further process steps.

For consumers (see upper lane) these are process steps, which enable formulating demand (*Complex product request*) and acquiring an overview of potential providers who might satisfy such demand (*Provider list*). For the providers (bottom lane) these are process steps to enable describing and registering offerings (*Product/service registration*). The resulting information objects (*Complex product request/Provider list*) enable consumers to reach potential providers and thus start the interaction processes of the following phases:

- The interaction processes in the intention phase, are characterized by several iterations between consumers and potential providers required to negotiate a preliminary agreement. Consumers initiate the negotiation processes and run them as long as at least one proposal is created (*Complex product proposal*) fitted to satisfy the requested consumer's demand.
- The interaction processes related to the contract phase, are defined by interactions necessary for creating and confirming a legally binding contract (*Complex product contract*) based on the preliminary agreement. As with intention phase, also here, the interactions are initiated by consumers leading the process of creating an umbrella contract based on confirmations received from the involved providers and ordering.
- The interaction processes in the settlement phase are characterized by interactions required for the fulfillment of the legally binding contract agreed in the previous phase. These fulfillment-related processes generate the transaction data and together with data generated in previous phases, they build a transactional dataset (*Transaction record*). The generated transaction records serve as the basis for all subsequent processes that might occur during the follow-up phase. Moreover, these are records of the institutional history, and are considered essential for building trust and reliability among DMS as a self-organized exchange environment.

After overviewing the interaction process model at a higher level, the following sections provide more detailed descriptions of individual phase-related processes. Section 4.6.1 focuses on processes essential for establishing a DMS as an exchange environment, such as the processes for joining and becoming DMS member and processes for the description of demand and supply. Section 4.6.2, Section 4.6.3, Section 4.6.4 describe processes, which enable the negotiation, contracting and settlement of complex products. This is followed by an overview of possible aftersales and knowledge sharing processes, and the description of a dispute resolution process in Section 4.6.5.

4.6.1 Knowledge Phase

Three main processes define the knowledge phase. As indicated in Figure 4.9 these are processes to enable:

- Joining and becoming a member of the DMS ecosystem (all roles)
- Describing and registering offerings (provider role)
- Formulating complex product requests and matching for potential providers (consumer role)

Unlike exchange environments with a centrally provided infrastructure, the DMS ecosystem is an open and self-organized environment of different actors taking different roles. In such self-organized networks, each member is at the same time the contributor of the underlying infrastructure and participant in an exchange market built upon this infrastructure. Therefore, the first step of further members is to register and share in which role (or roles) they intend to contribute to the ecosystem.

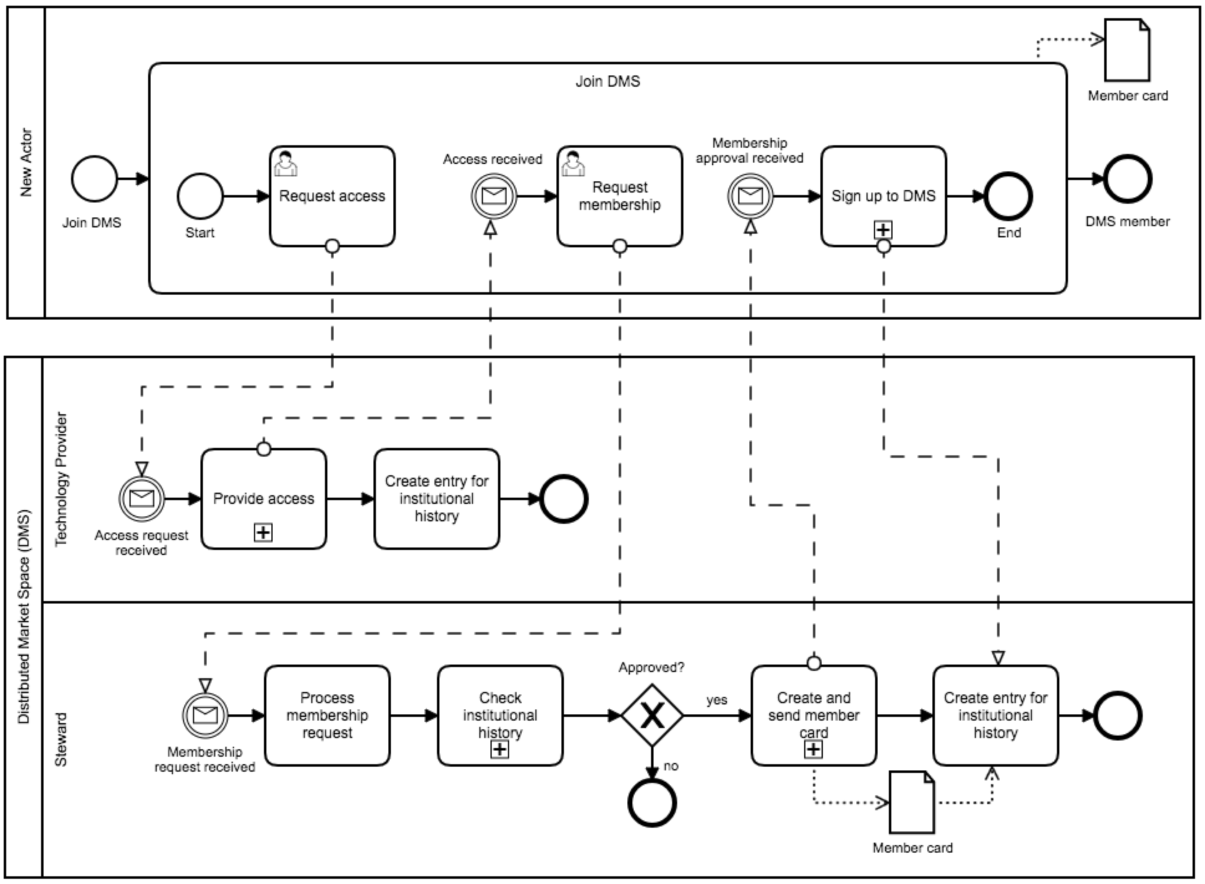

New Actor
Join DMS
Join DMS
Start
Request access
Access received
Request membership
Membership approval received
Sign up to DMS
End
Member card
DMS member
Distributed Market Space (DMS)
Technology Provider
Access request received
Provide access
Create entry for institutional history
Steward
Membership request received
Process membership request
Check institutional history
Approved?
yes
no
Create and send member card
Create entry for institutional history
Member card

Figure 4.10: Joining and becoming a member of the DMS ecosystem – a process describing a set of activities and interactions between a new actor and enabler roles technology provider and steward, necessary for becoming a DMS member.

Figure 4.10 shows the process of joining and becoming a member of the DMS ecosystem. The process starts with requesting access as indicated in the upper lane. The access represents an entry-point or link that enables initial interactions with the DMS (e.g., a website or link for download of a required DMS User Interface). Such access requests are processed by technology providers as indicated in the middle lane. Technology providers are actors whose primary responsibility is to provide software/hardware resources necessary for establishing and functioning of the ecosystem (cf. Section 4.5.2). After receiving DMS access, new actors request membership by sending membership requests, which are received by stewards. Stewards (i.e., actors responsible for the governance-related tasks) decide based on the defined rules and standards and existing entries in the institutional history. The institutional history of the DMS ecosystem represents a shared record of all registered members, their roles and relevant events and transaction-related data. Together with rules and standards, it is an instrument for decentralized governance of a self-organized network and builds the foundation for building governance services necessary for the growth and sustainability of the DMS ecosystem. Section 4.7.2 describes these services in more detail.

The membership request can either be denied or approved. The latter creates member cards (with unique identifiers) and sends them back to the requestors making an entry in the institutional history. After receiving member cards, new actors can sign-up to DMS, and register for different roles. Depending on the chosen role (or roles) the sign-up process might encompass further interactions with technology providers like, e.g., access to other tools or services.

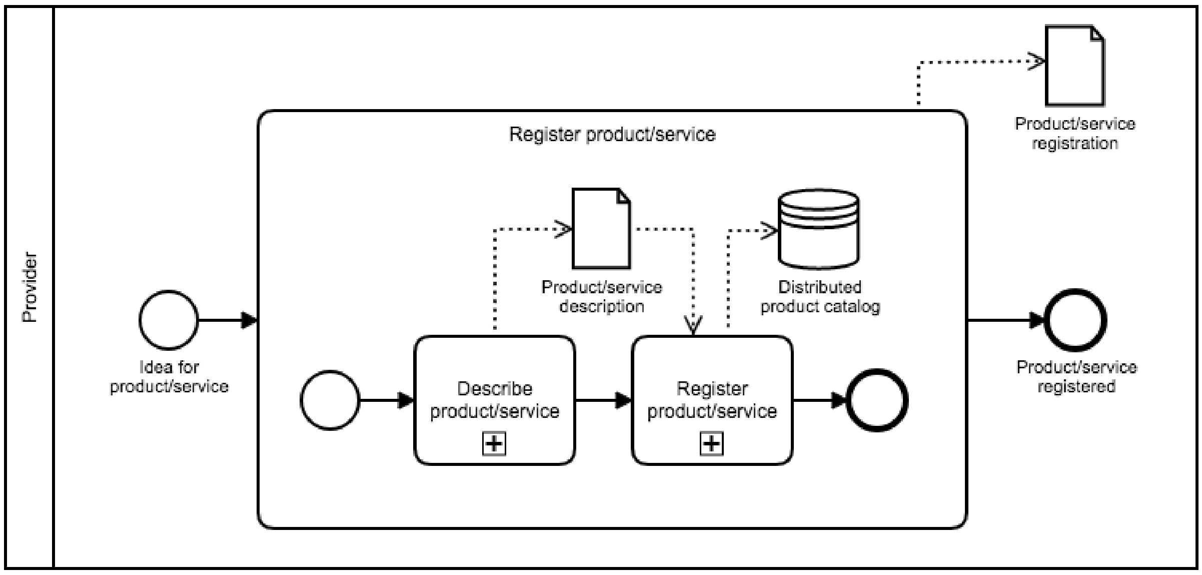

Provider
Register product/service
Product/service registration
Product/service description
Distributed product catalog
Idea for product/service
Describe product/service
Register product/service
Product/service registered

Figure 4.11: Register offerings – a process describing provider's activities with the objective to register a product or service to the DMS

After joining the DMS, members taking a provider role, start the process by describing their offerings as shown in Figure 4.11. Offerings, structured product/service descriptions, as well as short description of providers themselves (e.g., name, address, settlement modalities or ratings), are registered in a distributed product catalog. The distributed product catalog follows the same principle as the DMS institutional history. It serves as an instrument for publishing offerings among a trusted network of actors rather than in a mediated product catalog as the case with centrally orchestrated solutions.

On the other side, members taking the consumer role, start the process by formulating their demand. As shown in Figure 4.12, consumers are enabled to create complex product requests and match them for potential providers. The matching is done based on the data stored in the distributed product catalog. In case there are no matching results (i.e., no providers who can offer for the requested product/service combination), consumers need to modify their requests or leave the process. Otherwise, when matching results exist, a list of providers is generated. The generated provider list contains

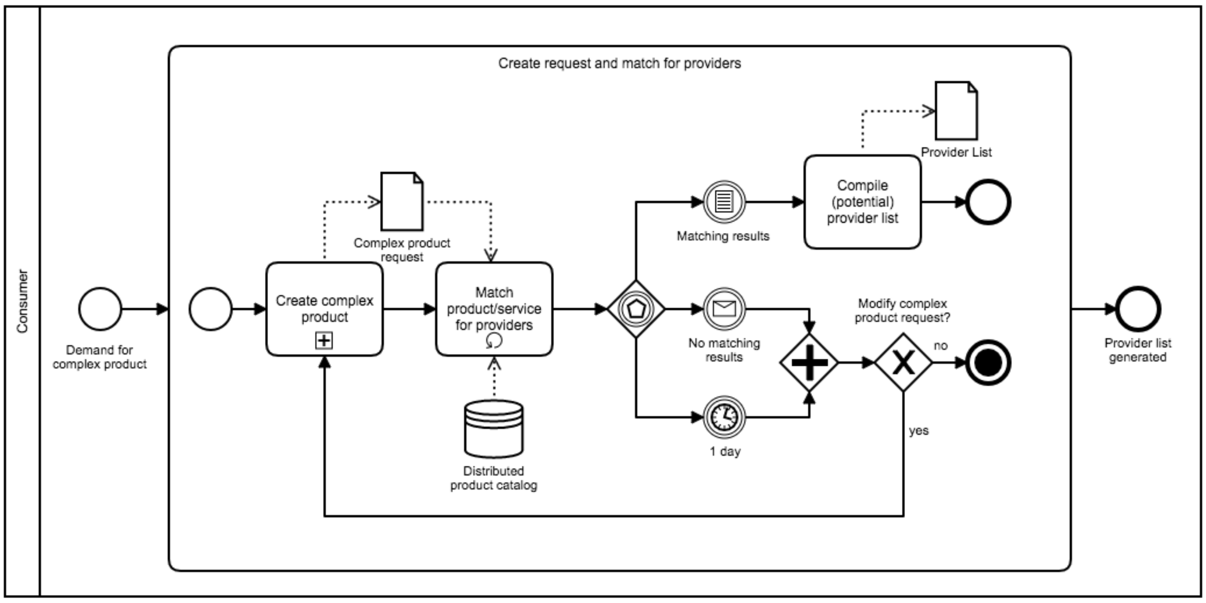

Consumer
Create request and match for providers
Demand for complex product
Create complex product
Complex product request
Match product/service for providers
Distributed product catalog
Matching results
Compile (potential) provider list
Provider List
No matching results
1 day
Modify complex product request?
no
yes
Provider list generated

Figure 4.12: Formulating a complex product request and matching for potential providers – a process describing consumer's activities with the objective of getting an overview of potential providers for a particular complex request

all necessary data required for addressing identified providers and starting interaction processes related to the following intention phase.

4.6.2 Intention Phase

After formulating demand and acquiring an overview of potential providers, consumers are enabled to initiate the intention phase. As shown in Figure 1.13, consumers start the process by addressing identified providers and sending them requests for offers (RfO). RfOs are requests for offers for individual parts of the complex product sent to providers. By sending RfOs, consumers signal their intention to engage in commercial exchanges with these providers. The contacted providers, on the other side, indicate their intention to provide for a particular product or service by answering RfOs and sending back concrete offerings (see Figure 1.13, bottom lane).

Complex products might get arbitrarily complex and include different product/service combinations. As a result, providers for each of the requested products or services need to be addressed in order to obtain offerings for each of them. The related activities (i.e., requesting offerings on the consumer side and creating and sending offerings on the provider side) comprise many interactions, which might undergo multiple iterations. Such interactions might repeat until enough offers are obtained and assessed hence enough viable offerings can be collected.

Collected offerings enable consumers to create complex product proposals and rank them based on their requirements and preferences. As a result, a best-fit list is created

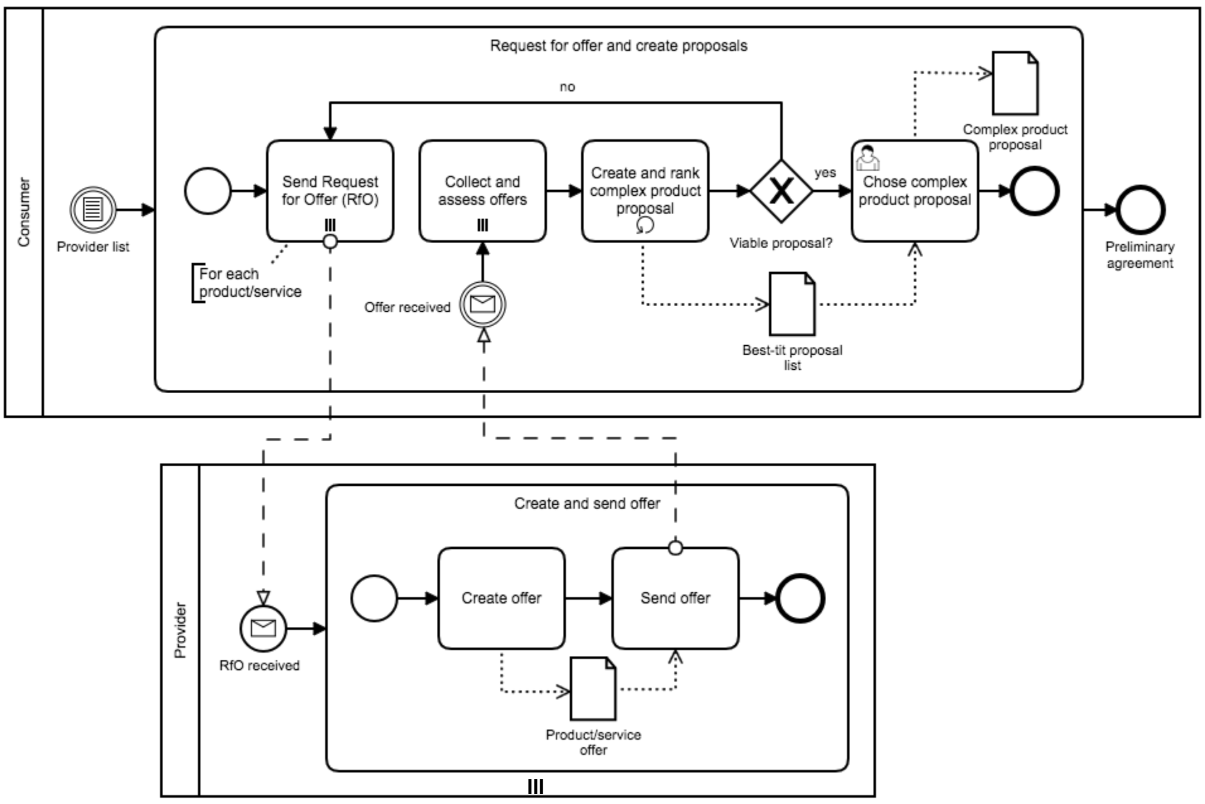


Figure 4.13: Requesting and receiving offerings, and creating viable complex product proposals – a process describing negotiation activities between consumers and providers, with the objective to achieve a preliminary agreement

to support consumer's decision making. In case there are no viable complex proposals, consumers might go back to the beginning and start another negotiation process undergoing the activities of sending modified RfOs and waiting for providers' response. Alternatively and in case there are viable proposals, they might choose one to proceed. The negotiating process between involved parties ends with a preliminary agreement on a selected proposal for a particular complex product.

4.6.3 Contract Phase

After accepting viable complex product proposals, consumers are enabled to enter into contractual agreements with the involved providers and setting up legally binding contracts.

Figure 4.14 shows the process of setting up a complex product contract. From the consumer side, a complex product contract is considered an umbrella contract since it might incorporate different arrangements for different parts. Such an umbrella contract represents a one-to-many contract situation and requires consumer's involvement in multiple contractual processes; one for each of the requested products or services in

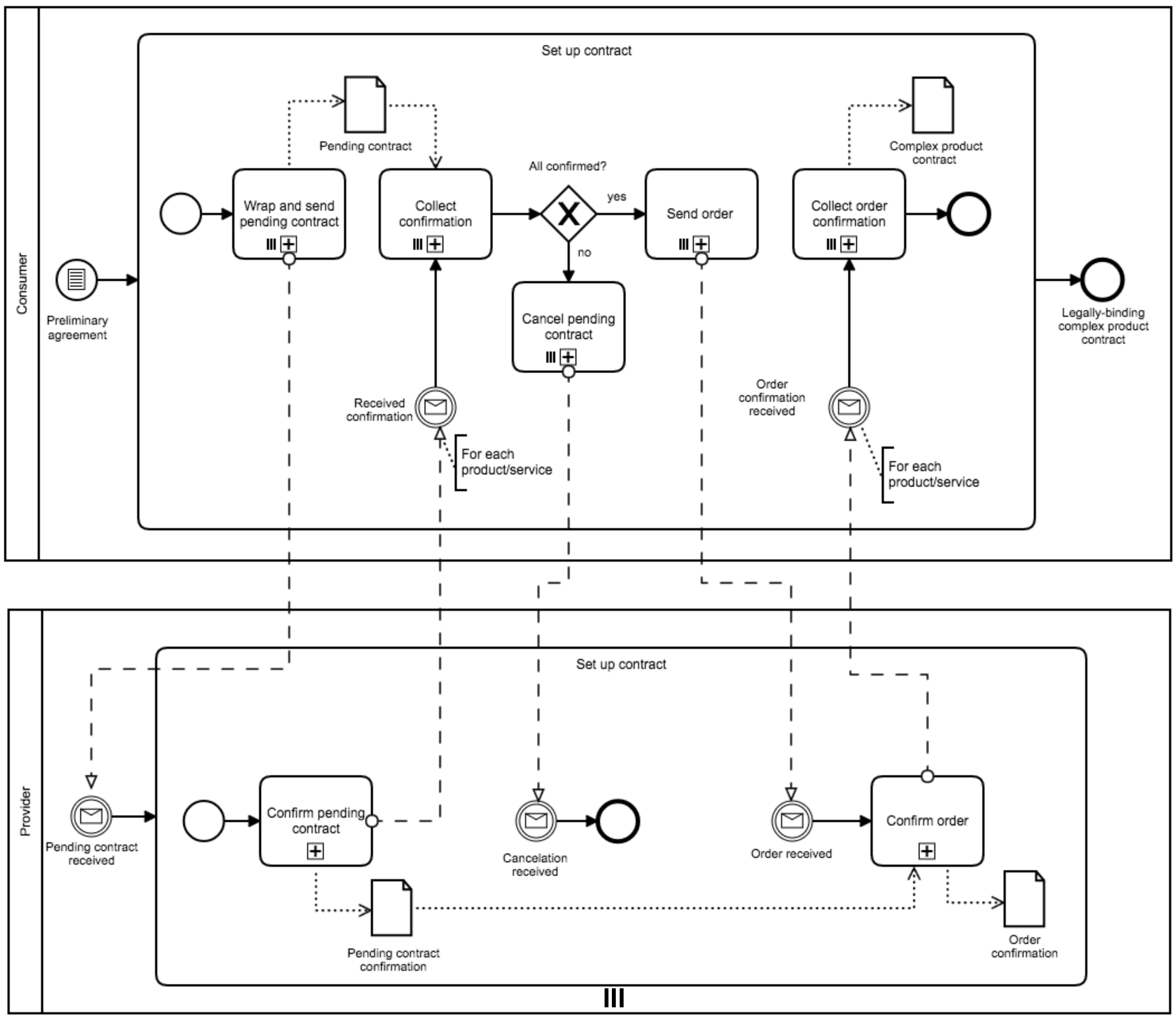


Figure 4.14: Setting up contract – a process describing contracting activities between consumers and providers, with the objective of agreeing on a legally binding complex product contract

order to achieve a contractual agreement with all involved providers (counterparts) for all required products or services.

The process of setting up an umbrella contract is two-staged. The stage one is considered provider-confirmed and stage two consumer-confirmed. In the provider-confirmed stage, a pending contract for each of the required products or services is created and sent to the involved providers for confirmation. Only if all contacted providers confirm all pending contracts, can the second confirmation stage start. The consumer-confirmed stage begins with placing orders for provider-confirmed (pending) contracts and receiving order confirmations. After all order confirmations are received, an umbrella contract is created to bring together all confirmations.

The supporting activities for the two-stage confirmation process of a complex product contract are ranging from creating pending contracts, sending them to providers, collecting confirmations and finally placing orders (see Figure 4.14, the upper-lane). On the provider side, the activities related to contracting are slightly different and considerably more straightforward. That is because the contracting process on the provider side is considered a one-to-one contract situation with an additional activity encompassing the confirmation of pending contracts as the prerequisite to the final order confirmation. The setting up process ends with a legally binding complex product contract. As with all legally binding contracts, it is mandatory for all counterparts and entails all terms and conditions required for its settlement and enforcement.

4.6.4 Settlement Phase

After setting up the contractual agreement for complex products, transactional parties are enabled to fulfill obligations resulting from that contractual agreement. These are payments on the consumer side and delivery of ordered products and services on the provider side. Such settlement processes are well researched and understood (see [Kol13], [PRW03] and [SS98]). The main issue with settlement processes between transaction partners who do not know each other is the lack of trust that each of them will fulfill the contractual agreement. Centralized models compensate such trust issues by centrally organizing settlement processes, and by doing so position themselves as the trusted intermediary.

Against this background and depending on the expected level of trust, a reference DMS suggests three different approaches regarding the settlement of complex products:

- Trusted settlement
- Trusted third-party settlement
- Trustless settlement

Trusted settlement assumes an adequate level of trust among transaction partners within a reference DMS. That might be the case in situations where consumers know and trust their providers, or, they are involved in complex product scenarios perceived as a low-risk scenario. Take for example the use case scenario of a couple looking for a bundle of personalized services (theatre tickets, babysitter, a table at a restaurant and reserving a parking spot) (cf. Section 3.3.). The settlement of standardized products and services from the low-mid price segment offered by established providers (e.g., Eventim for tickets or MyTable for restaurant reservation) is usually perceived to be lower risk, as in the case of personalized products and services from the premium segment.

Figure 4.15 illustrates the process of fulfilling a complex product contract by applying the trusted settlement approach. On the consumer side, the process starts with the payment for each product and service, and sending a payment notification to each of the involved providers. After receipt of payment, the process on the provider side starts with sending back the receipt of payment followed by the delivery of the order. The order delivery comprises physical delivery (e.g., provision of a parking spot), provision

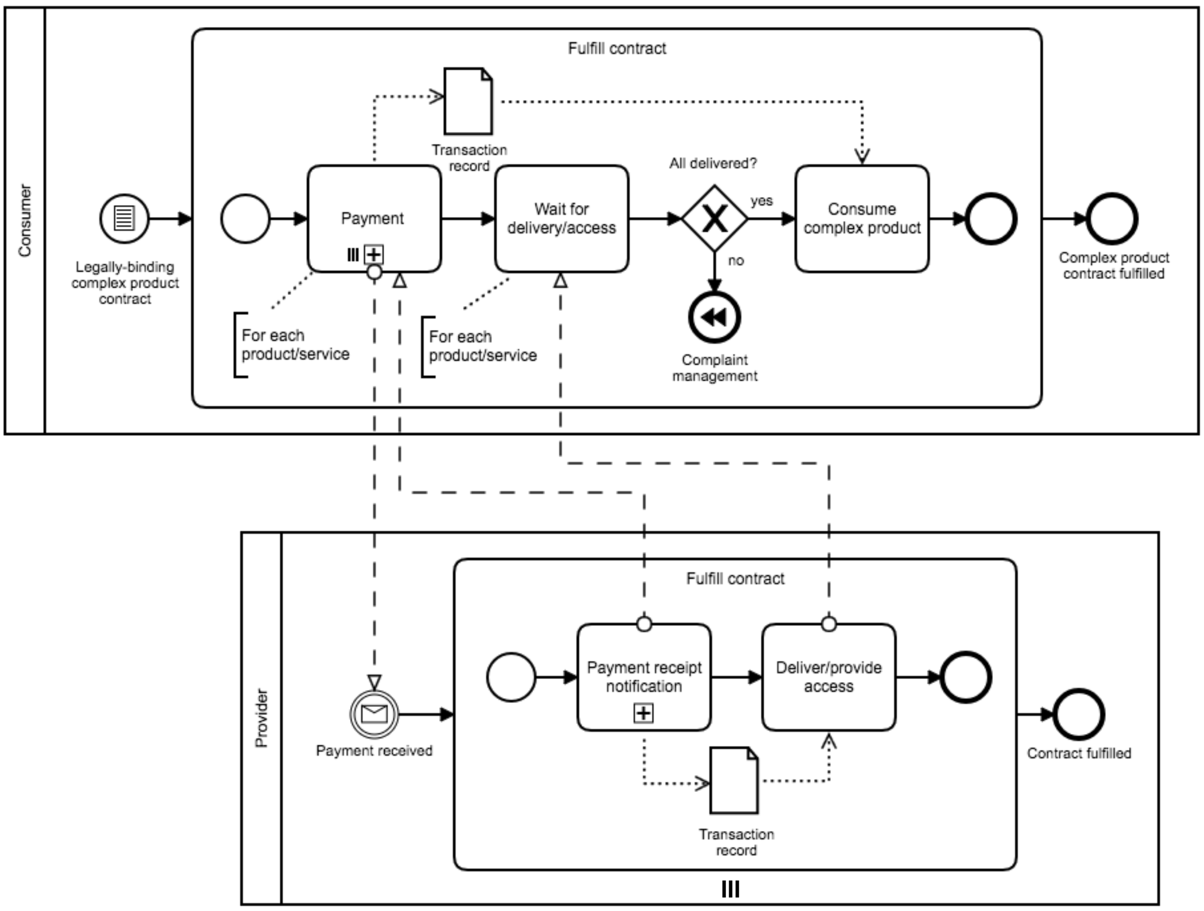


Figure 4.15: Fulfilling complex product contracts – a process describing interactions among transaction parties by applying the trusted settlement approach

of access (e.g., theatre tickets accessed by QR codes), as is the case with digitally available products or services. On the consumer side, the process continues with waiting for delivery of all orders. In case of delays or other unexpected events, consumers might ask for compensation and alternatively initiate complaint management. Otherwise, the consumption starts after delivery of all orders, and the whole process ends with creating a transaction record and an entry for the institutional history.

Trusted third-party settlement assumes engaging a third party trusted to be capable of ensuring safe contract fulfillment (i.e., payment and delivery according to the contractual agreement). In the DMS ecosystem, trusted third parties might come from the ecosystem's participants taking the mediator or expert roles as described in Section 1.3. Mediators or experts may be involved in the settlement process and support consumers and providers to fulfill obligations defined by the agreed contract. This approach is considered suitable for settlement of product/service combinations from the higher price segment and higher complexity regarding coordination and management of payment and delivery activities. Take for example the use case of a house owner who is looking to renovate his bathroom, with the objective to stay within a tightly defined budget and schedule (cf. Section 3.X). /todochange!

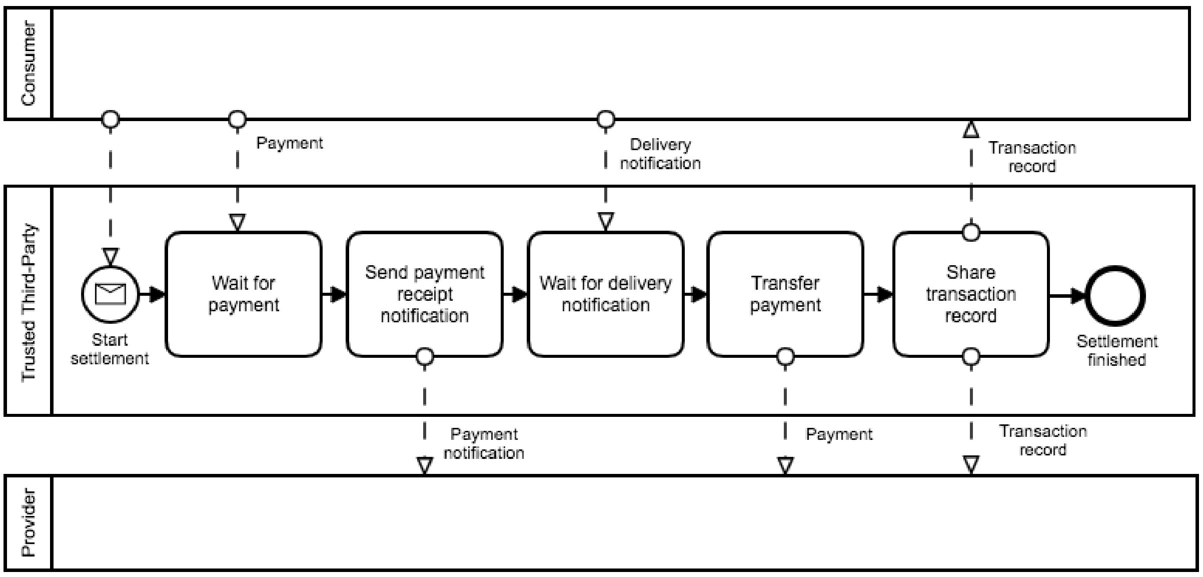


Figure 4.16: Fulfilling complex product contract – a collaboration diagram summarizing interactions between transaction partners (consumers and providers) with a third party applying the trusted third-party settlement approach

Figure 4.16 shows the process of fulfilling a complex product contract by applying the trusted third-party settlement approach. It summarizes the collaboration of involved consumers and providers with a mediator as the trusted party. The main activities of the involved mediator (see middle lane) are to coordinate the payment process according to the agreed terms and conditions. The coordination process starts with the payment of the consumer side as the signal of the willingness to pay for the order. After receiving a payment notification from the mediator, providers deliver ordered products and services (i.e., provide access to them). Following the delivery notification from the consumer side, the mediator transfers payment to the providers, and hence closes the payment transactions for the whole complex product. The settlement process ends with the sharing of the created transaction record.

The *trustless settlement* approach builds on the concept of smart contracts. Smart contracts are automatable and enforceable contractual agreements, which could be implemented as immutable computer programs to run on a broader range of technology platforms, including distributed ledger platforms such Ethereum [6] and Fabric [11]. Smart contracts are ”trustless” since they refer to a ”piece of computer code” that executes the terms of a contract, and by doing so minimizes the need for trust between transaction counterparts or the need for a trusted third party [Sza97]. By using smart contracts, the contractual clauses (i.e., terms and conditions) and interactions between transaction parties are translated into code and are transparent and self-enforced. In case any party deviates from the contractual agreement the following actions, e.g., payment and penalty, are known and automatically enforced by the smart contract. Even though smart contracts and distributed ledger technology are still in their early stages

[Luu+16],[AW18], the trustless approach seems promising for the settlement of complex products, in particular for IoT scenarios. As described earlier in Section 3.2, such scenarios incorporate predominately digital products and services, and require many different and partly heterogeneous providers (e.g., autonomous agents and machines) to engage in order for the complex product to realize.

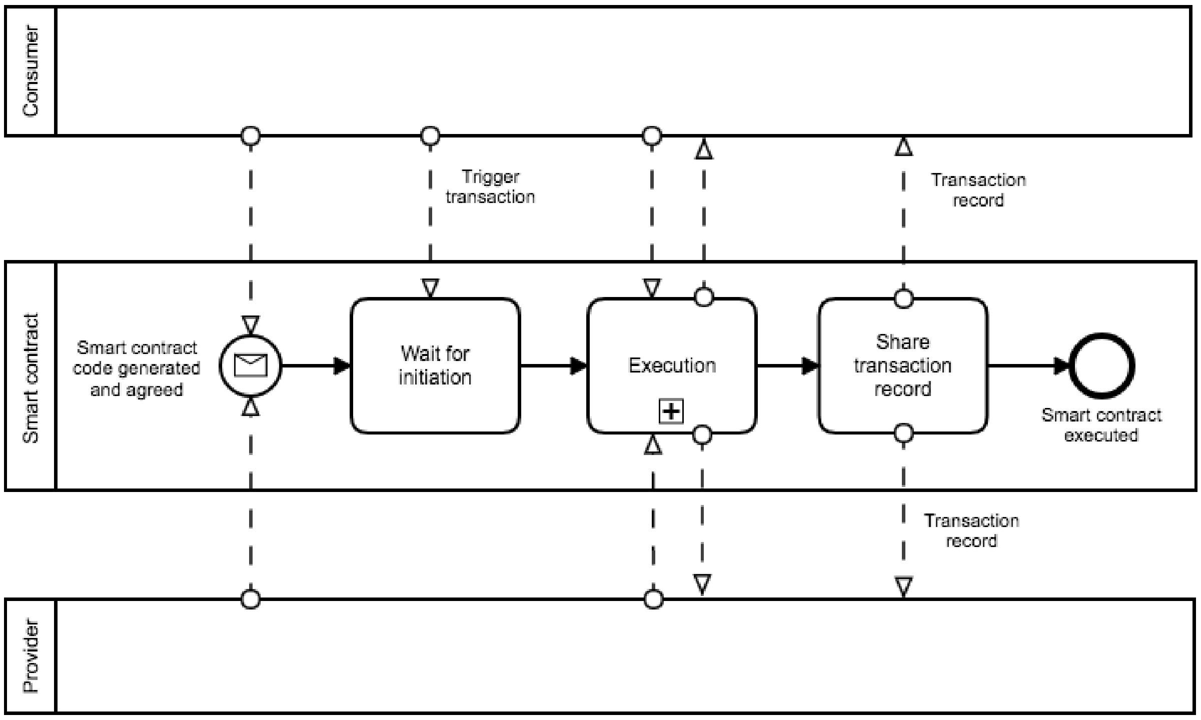


Figure 4.17: Fulfilling complex product contract – a collaboration diagram summarizing activities related to the initiation and execution of smart contract representing the trustless settlement approach

Figure 4.17 illustrates a simplified process of trustless contract settlement based on smart contracts. As shown, transaction partners (consumers and providers) first need to generate the smart contract code that entails the terms and execution logic for the legally binding complex product contract. After all involved partners agree on the generated code, it is deployed in the underlying blockchain waiting to be triggered by the consumer. Once the smart contract is initiated, it executes itself. As a result of the process, a transaction record is created and shared with involved parties.

4.6.5 Follow-Up Phase

The processes of the follow-up phase close the market transaction for complex products. As indicated in Figure 4.9 it encompasses processes to enable:

- After-sales processes
- Dispute resolution

After-sales processes for complex products are considered the same as for individual products and services. These are processes that enable transaction partners to review settled transactions, to extend their relationship by offering additional activities like for example, customer support, as well as to handle return and refund [SS98], [Kol13]. Reviewing settled transactions primarily includes giving two-sided reviews by involved parties. Moreover, the reviewing process might combine sharing experiences with other ecosystem participants as well as sharing transactional data. Based on that, reviews are used to support decision-making considering market transactions, but also as an indication of the ecosystem's ability to support market transactions. For a detailed description of the aforementioned aftersales processes see [Fra04].

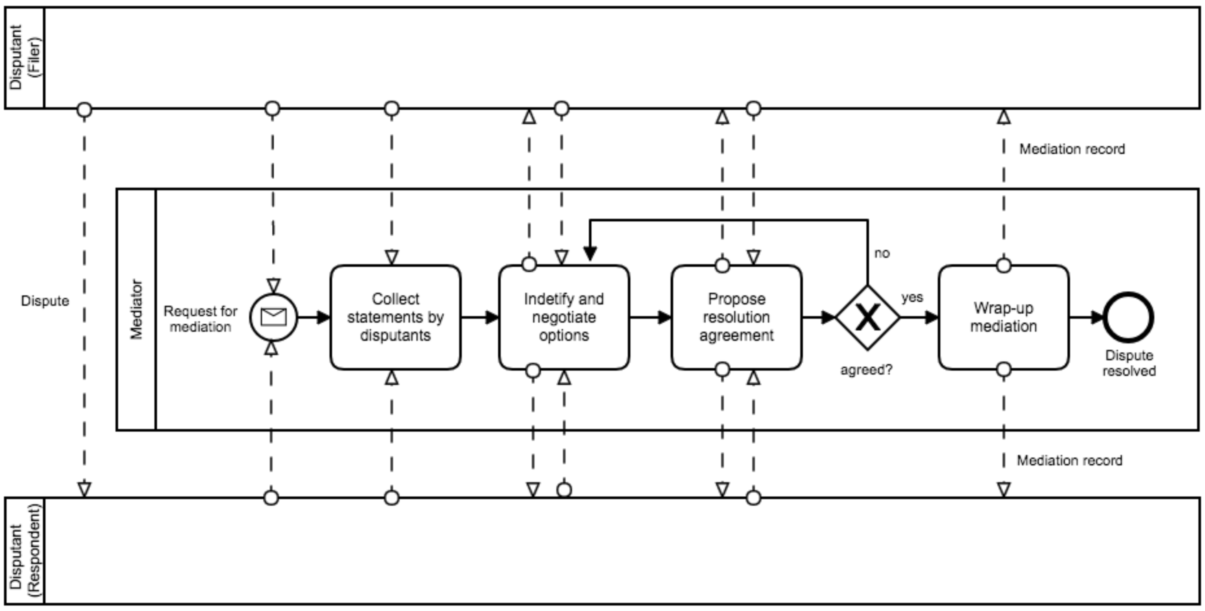


Figure 4.18: Resolving a dispute – a collaboration diagram summarizing interactions between involved parties (disputants and a mediator) necessary for achieving a resolution agreement

Dispute resolution processes are needed to enable DMS participants to settle potential disputes. A dispute is considered a form of a conflict in which one party makes a claim (called filer), and the other party rejects that claim (called respondent) [FAS80]. In recent years, diverse Online Dispute Resolution (ODR [KKR01]) approaches were developed following, in general, the same process. Accordingly, an online dispute resolution process must allow disputants to choose an adjudicator, i.e., a mediator, who is capable of resolving such issues. The mediator, on the other hand, is required to be as neutral as possible and contribute to the dispute resolution in an efficient and swift manner.

Following these recommendations Figure 4.18 presents a exemplary process for dispute resolution within the DMS ecosystem. It illustrates relevant interactions between disputants (a filer and respondent) and a mediator who intervenes with the objective to resolve the dispute in place. In disputes related to complex products, the filer is normally the consumer, and the respondent is usually the provider. The mediator is a DMS participant taking the role of an expert or mediator (cf. Section 3.1). The dispute resolution starts with the recognition of a dispute and the willingness of the filer and the respondent to appoint a mediator. After choosing a trusted mediator, the actual resolution process starts with the request for mediation by involved parties. After accepting the mediation mandate, the mediator organizes hearing sessions to collect dispute-statements by all involved actors, followed by identifying and negotiating viable options. As a result, a binding resolution agreement is proposed by the mediator and in the best case agreed by disputants. The process ends by wrapping-up mediation records and sharing them among disputants.

In conclusion it can be noted that aftersales processes are predominately related to the follow-up phase. However, they also might take place as a parallel process in other phases, as well as, intersect several phases. In particular, this is the case with sharing transactional data, which can be involved in different activities and as part of various process steps. Bearing this in mind the next section will discuss the above-described processes in the context of services, considered necessary for their implementation in a reference DMS.

4.7 Service View

The service view defines services that a reference DMS must provide in order to support the ecosystem's organizational structure and its core interactions. Identified services build a *Service Stack* required to implement the previously introduced ecosystem model (cf. Section 4.5) and related interaction process model (cf. Section 4.6).

Figure 4.19 illustrates the proposed *Service Stack*. It summarizes the identified services, grouping them according to their functions or their affiliation to a specific aspect of a reference DMS. It comprises three groups of services:

- *Foundational Services* are necessary for the establishment and functioning of the DMS ecosystem. This group of services enables direct communication and interactions among DMS participants directly and reliably.
- *Governance Services* are necessary for upholding the DMS ecosystem. This service group increases value and enhances the growth of the DMS ecosystem, and contributes to its capability to guard against misbehavior and outright fraud.

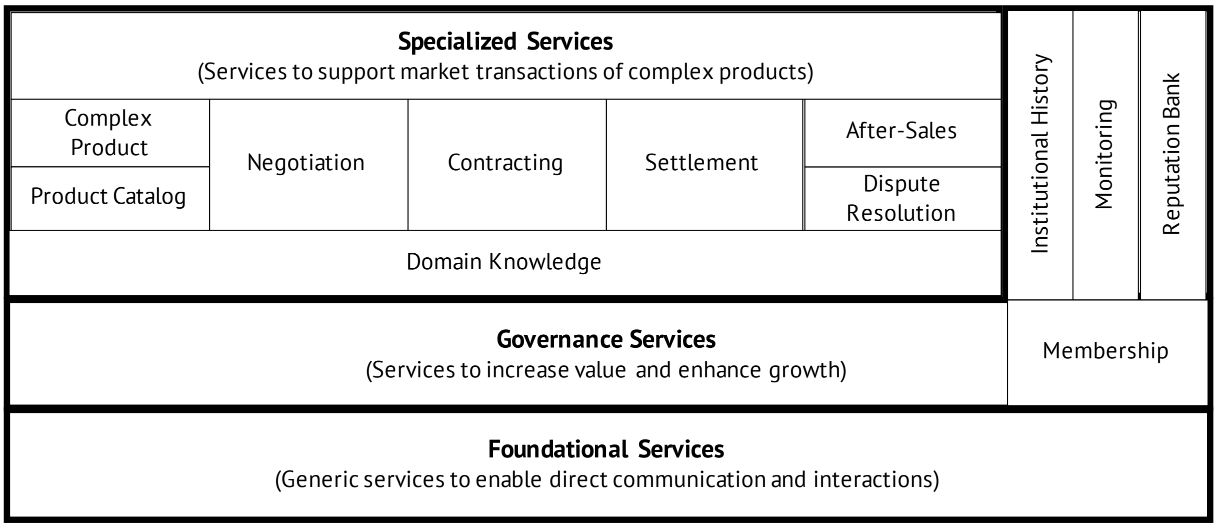


Figure 4.19: Service Stack of a reference DMS

- *Specialized Services* are necessary for facilitating the market exchange within the DMS ecosystem. As a group of eight services, they implement the interaction process model by enabling distinct process steps related to different phases of market transactions for complex products.

As indicated in Figure 4.19 service groups are linked together in a hierarchical order (i.e., a stack), whereby the foundational services form the base of the DMS service stack. Based on that, governance services provide a self-governance framework that sets the desired behavior for the DMS ecosystem to which specialized services must adhere. This ensures that specialized services follow the rules and norms that are implemented and monitored by governance services. Together, these services contribute to the ecosystem's capability to develop, sustain and provide for its health and vitality.

4.7.1 Foundational Services

Foundational services support the establishment and functioning of the DMS ecosystem by enabling direct communication and interaction between its participants. As a self-organized and decentralized environment, a reference DMS builds upon a network of participants who are equal in their rights and responsibilities. Such networks of equal participants (also referred to as peer-to-peer networks) use P2P overlays as their underlying communication and collaboration infrastructure. Considering the main functionality of P2P networks (cf. Section 2.2), the primary responsibilities of foundational services are, therefore:

- to enable discovery, locating, routing and messaging among ecosystem's participants, and

- to ensure the security and reliability of the ecosystem's communication infrastructure.

Discovery services facilitate finding other participants within the ecosystem. *Locating and routing services* optimize the path of a message traveling from one DMS participant to another. *Messaging services*, on the other hand, are critical to support direct interactions between participants. They enable addressing participants of interest and sending them defined messages.

Security and reliability services address two additional inherent challenges for the DMS as a decentralized, networked environment. While the security service is required to ensure that only authorized participants (i.e., DMS members) have access to information provided by other participants, the reliability service is expected to guarantee a reliable behavior within the ecosystem as a whole.

Even though considered generic, foundational services might differ in their characteristics and performance. This is subject to the concrete implementation of the underlying P2P overlay and applied mechanisms and algorithms. As further explained in Chapter 5, this work uses CHORD [Sto+01] as the P2P overlay for the prototypical implementation of the DMS Architecture.

4.7.2 Governance Services

Governance services constitute the governance structure of a reference DMS as a self-governed environment with no central instance of the control. They increase value and growth and help protect the ecosystem from misconduct and outright fraud.

Following the design principles for self-governed communities proposed by Ostrom [Ost15] and the peer production appoach by Benkler [Ben06], the following services outline the self-governance structure of a reference DMS:

- Membership
- Monitoring
- Institutional History
- Reputation Bank

Membership Service is necessary to implement processes essential to forming the DMS ecosystem. This includes processes related to the joining (and leaving) the ecosystem as well as registering for different roles as presented in Section 4.6.1. Moreover, membership service is to enable transparency and clarity about who is performing which activities in which roles and how they align to ensure that participants follow the ecosystem's code of conduct defined by rules and norms.

Monitoring Service is required to help regulate behavior among the DMS ecosystem, which are described by rules and norms. Thus rules are considered explicit rules that define for example the term of participation or service, and norms refer to ways of behaving expected to align by all participants. Monitoring service enables scanning for ”rule-breakers” and supports their enforcement. Besides, it integrates tools and services to facilitate activities of the steward role (cf. Section 4.5.2). These are for example creating an easily accessible and well-structured documentation about the code of conduct, communication of relevant issues and events, as well as creating open records to explain their decisions and make them transparent for the rest of the ecosystem.

Institutional History Service is necessary for the logging of events relevant to the DMS ecosystem performance. It implements a distributed record of all registered members, their roles and related activities, which generate transactional data. As with many self-governed ecosystems (e.g., Wikipedia), the institutional history is considered a primary instrument to ensure the ecosystem's internal transparency. The institutional transparency is seen as essential in order to help members use existing and develop new resources [PVC16]. Institutional transparency fosters trust among participants, since trust is not a static concept and grows over time as a result of experiences and interactions within the ecosystem.

Reputation Bank Service is required for the creation and maintenance of a reputation bank that holds records about the ecosystem's members regarding their reliability, solvency, and worthiness. It implements a distributed record that on the one hand captures two-sided reviews about settled transactions as an indication of reliability and solvency. On the other, it may keep records about social value (i.e., a subjective form of value) upon which members build their excellent behavior and contributions to the community. The rationale behind this concept coined as 'social currency' [McC14] is to reward a desired behavior that counts for the assessment of the member's reputation. For example, members taking the expert or mediator role might collect credits by providing excellent services to other members, or stewards can receive credits for their commitment and contribution for the good of the ecosystem.

4.7.3 Specialized Services

Specialized services facilitate market exchange within a reference DMS by supporting processes related to market transactions of complex products, and thus implement the interaction process model presented in Section 4.6.

As indicated in Figure 4.20, eight services are necessary to implement each of the five phases of the process model. These are as follows:

- Complex Products
- Product Catalog
- Negotiation
- Contracting
- Settlement
- Aftersales

- Dispute Resolution
- Domain Knowledge

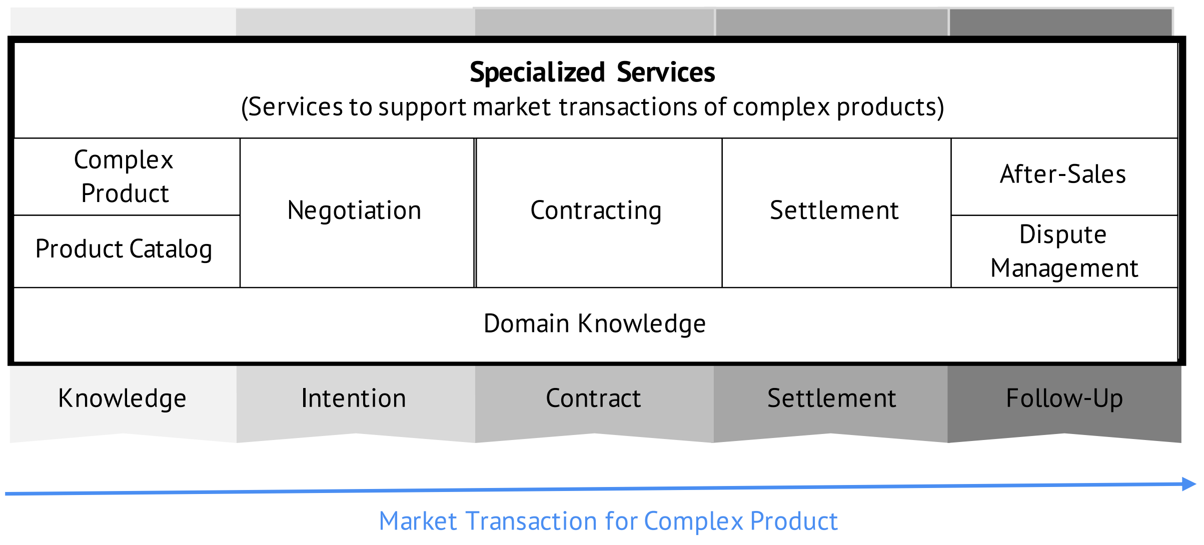


Figure 4.20: Specialized services related to the phase model of market transactions for complex products

Complex Product Service supports consumers to formulate their demand and acquire an overview of potential providers for a particular complex request. It implements a set of tools and services that seamlessly support different process steps as described in Section 4.6.1 (see Figure 4.12).

Product Catalog Service enables the creation and maintenance of the product catalog of products and services offered via the DMS. It assists providers in describing and registering their offers, hence implements the process illustrated in Figure 4.11.

Negotiation Service implements interaction processes related to the intention phase (cf. Section 4.6.2). It supports negotiation processes between consumers and providers helping them to achieve a preliminary agreement for a specific complex product as illustrated in Figure 4.13.

Contracting Service enables transaction partners to create legally binding contracts for complex products. The contracting service implements the process of creating umbrella contracts by supporting the two-stage contract confirmation process as described in Section 4.6.3 (see Figure 4.14).

Settlement Service supports the transaction partners in fulfilling contractual agreements regarding payment and delivery. The settlement service implements three different settlement approaches. Namely, trusted settlement (Figure 4.15), trusted third-party settlement (Figure 4.16) and trustless settlement (Figure 4.17).

Aftersales Service implements processes related to the coordination and management of return and refund, reviews and customer support. It also includes supporting transaction partners to review settled transactions by giving two-sided reviews and sharing their experiences with other participants to inform their decision-making (cf. Section 4.6.5).

Dispute Resolution Service implements the dispute resolution between DMS participants (cf. Section 4.6.4, see Figure 4.18). It facilitates involved parties achieving resolution agreements, but also integrates other participants such as mediators or experts (cf. Section 4.5.2), who might intervene as an adjudicator or an enforcer of such agreements.

Domain Knowledge Service enables the sharing of knowledge relevant for the market transactions in a particular domain (i.e., domain knowledge). Domain knowledge entails standardized ontologies and vocabularies, and other domain-related terms and conditions. Such domain knowledge might come from different sources, but it needs to be published by DMS members in order to be used by other services from the DMS service stack (e.g., product catalogue and complex product services). This is particularly the case with members taking the expert or knowledge provider roles who are responsible for providing reliable domain-knowledge (cf. Section 4.5.2).

In addition to the descriptions above, Chapter 5 provides further explanations of specialized services. Chapter 6 demonstrates their inner workings and shows how they, when linked with foundational and governance services, create value for DMS participants, and thus for the ecosystem as a whole.

4.8 Infrastructure View

The infrastructure view outlines the technical infrastructure of a reference DMS necessary for the implementation of the previously defined service stack (cf. Section 4.7). The technical infrastructure encompasses the groundwork for the implementation of foundational, governance and specialized services distinguishing between:

- Architectural and
- Technology-related aspects

4.8.1 Architectural Aspects

An information system needed to support a reference DMS as a self-organized and strictly decentralized market-oriented environment must follow the same design principle, since the system architecture should primarily relate to its function or purpose. As already explained in Chapter 3, the underlying system must use distributed resources according to the "form follows function" principle to implement the DMS service stack. The central architectural concern of such a system is to regard the actors involved as market participants who participate in a decentralized, i.e., peer-to-peer market and its mechanisms, and as constitutive parts that actively contribute to these mechanisms. Consequently, an underlying system for a reference DMS must be designed to integrate the above principles and concerns.

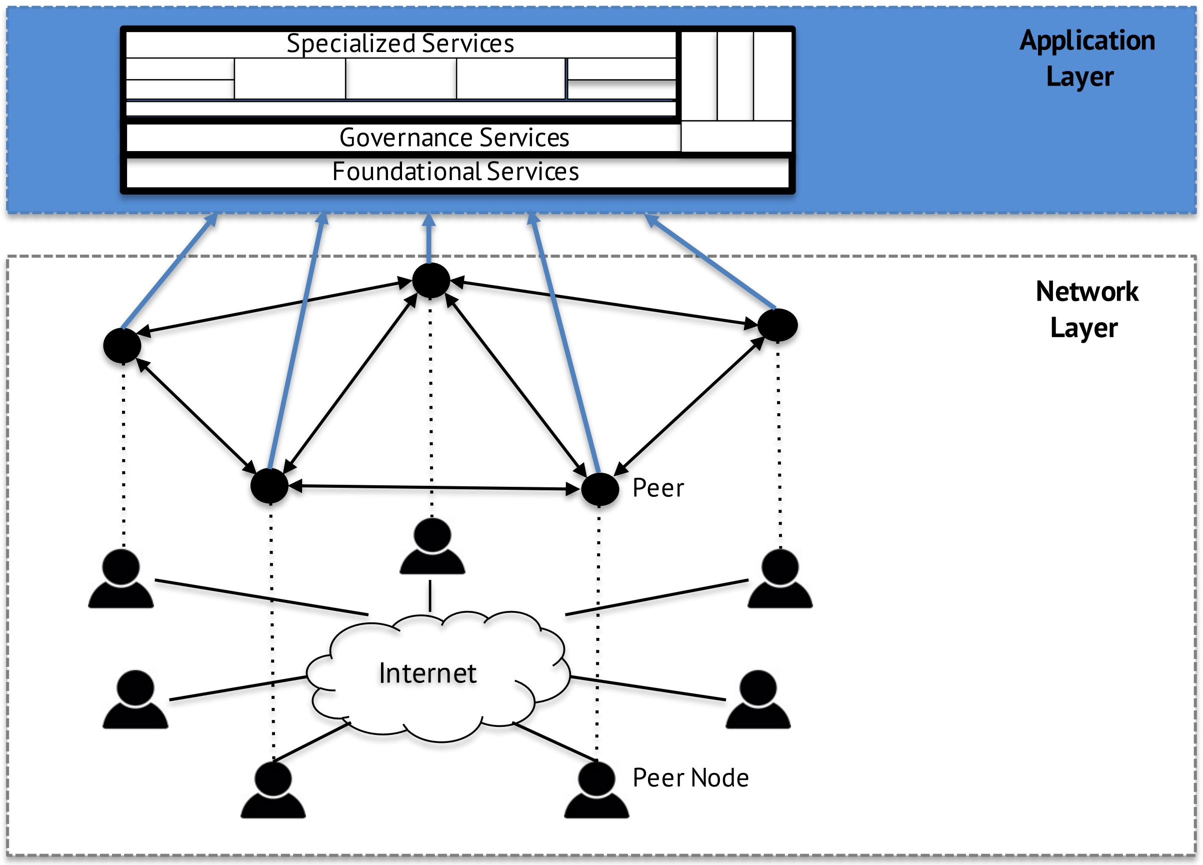


Figure 4.21: Technical infrastructure for a reference DMS – architectural aspects

Figure takes a high-level approach to such a system, showing its entities on the *application layer* and *network layer*. The *application layer* is necessary to enable governance and specialized services which need to be available to all actors, regardless of the role or roles they perform. It builds on the *network layer* represented by an overlay network, i.e., a virtual network composed of direct connections among peers on top of the Internet as the existing communication infrastructure (cf. Section 2.2.1). Peers are autonomous devices located at the edges of the Internet (i.e., computers, mobile devices or machines) owned by actors (actor node). They are communicating with other peers as equals as they consider to have the same rights and capabilities, and can take any of defined roles.

The application layer draws on the cooperation and collaboration of peers in the form of services (i.e., functional services), which peers provide to each other in order to achieve the implementation of the service stack, and thus, facilitate the organizational structure of a reference DMS. Such operative services (represented by directed blue lines) might either be provided resources or tasks to support specific services. For example, peers might provide resources for storing information, e.g., product catalog, or peers can provide tasks necessary for upholding the DMS governance structure, e.g., institutional history.

4.8.2 Technology-related Aspects

On the network layer, Internet and P2P Overlay Networks are considered essential for the communication and interaction infrastructure of a reference DMS (see Figure 4.22). This includes standards and protocols that cover a broad spectrum of communication paradigms since DMS actors can be everyone or everything connected to the Internet. For example, at one end of the spectrum are personal computers connected to the Internet, and on the other, e.g., devices and sensors connected in an ad-hoc manner via a wireless medium.

Furthermore, overlay networks integrate components necessary for the implementation of foundational services, and they are also responsible for enabling interactions and collaboration among DMS participants in a trustful and reliable manner. Even though different architectures for network overlays exist (e.g., Chord [Sto+01], Pastry [RD01], Tapestry [Zha+04]), the choice for a particular overlay network depends on the specific requirements of the application domain since their performance might differ very strongly. Notably, this is because of different applied mechanisms and algorithms and is also the subject of the concrete implementation.

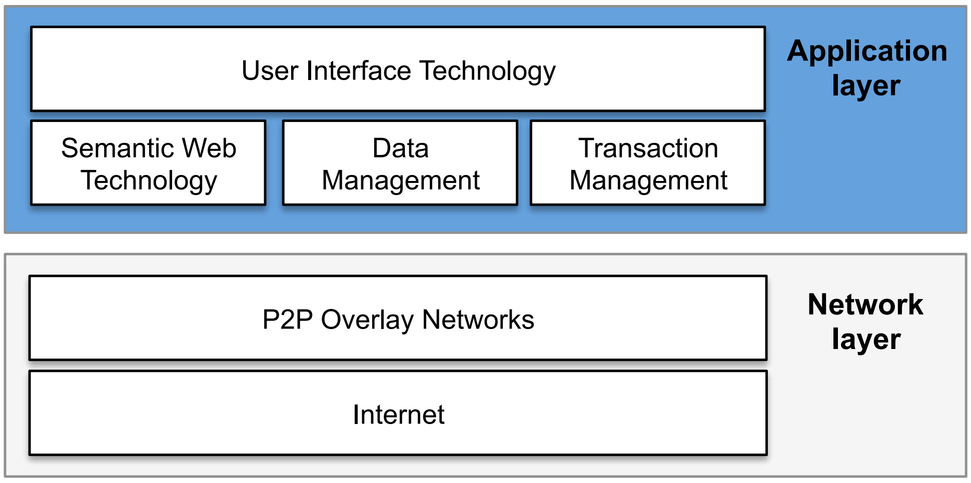


Figure 4.22: Technical infrastructure for a reference DMS – technology-related aspects

On the application layer, Semantic Web, Data Management, and Transaction Management technologies are considered necessary for the realization of governance and specialized services. Semantic Web [BHL01] and related standards (e.g., RDF 1.1 [Eri14]) is required in order to enable sharing and reusing rich data descriptions. This includes using existing ontologies for the structured description of knowledge relevant for commercial exchange in a domain of interest (e.g., Linked Open Commerce [KHB15]).

Data management is necessary for organizing, storing and retrieving information within a decentralized system. In particular this relates to databases designed to hold RDF data (i.e., RDF stores or RDF repositories), optimized for storing and finding RDF data using query languages like SPARQL [HS].

Transaction management technology is necessary for the realization of specialized services for example transaction settlement, considering different settlement approaches (e.g., smart contracts and distributed ledger technology [AW18]).

User Interface technology is considered essential for the implementation of flexible user interfaces to cope with complex products. Hence, user interfaces must support consumers seamlessly composing complex products and creating complex product requests. For this purpose, a novel concept of generic user interfaces (generic UIs [Hit+16] was developed and prototypically implemented as further described in Chapter 5.

4.9 Life Stages

This chapter focuses on the third dimension of the reference model for distributed market spaces. The *Life Stages Model* represents that dimension and covers three stages a distributed market space might undergo during its lifecycle *Design, Ignition*, and *Maturity*. It follows the principle of separation of concerns, where each of the life stages has different concerns and therefore different priorities, which in turn necessitate different activities in order to reach the threshold for the next stage.

Figure 4.23 presents the proposed life stages model for distributed market spaces and summarizes the concerns of each of the three stages as described in the following:

- *Design stage* – when a DMS instance is blueprinted, prototyped and launched
- *Ignition stage* – when a DMS instance is building the critical mass of participants and clearing frictions and bottlenecks
- *Maturity stage* – when a DMS instance is retaining existing participants (network) and connecting to others for sustainability and growth

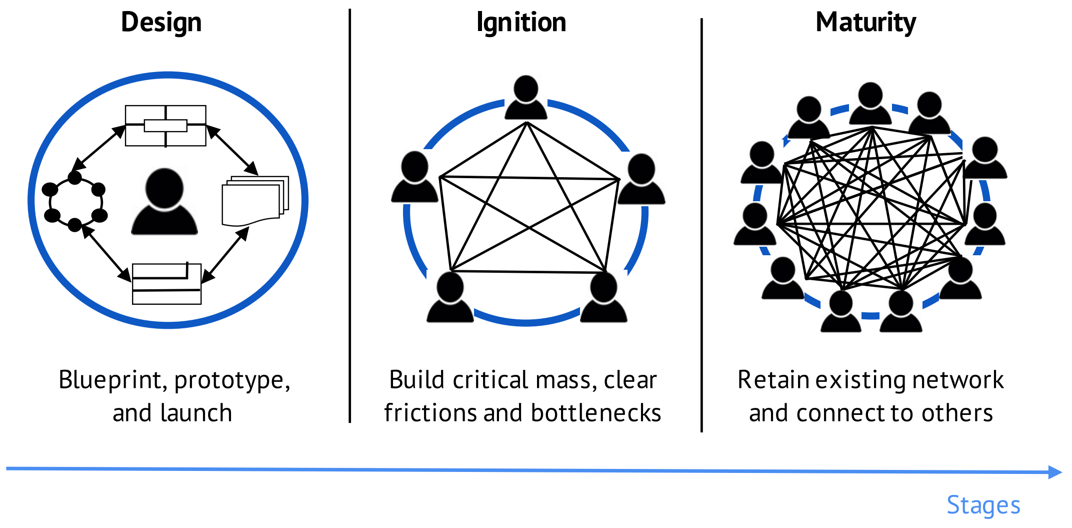

Design
Ignition
Maturity
Blueprint, prototype, and launch
Build critical mass, clear frictions and bottlenecks
Retain existing network and connect to others
Stages

Figure 4.23: Life stages model for distributed market spaces

4.9.1 Design Stage

The priority of the design stage is to proof the design hypothesis using a prototypical implementation for an application context. Therefore, *the primary concern is to blueprint, prototype, and launch an instance of a distributed market space that considers the contextual requirements of the particular application.* Blueprint is a conceptualization of an instance that considers different modeling views on the one hand, and integrates the application context on the other. As a result, a DMS blueprint comprises four extracted models as indicated in Figure 1.7.1. Together the extracted ecosystem model and accompanied process, service and infrastructure models shape the conceptual structure for the DMS instantiation in that particular application context.

Related activities: As summarized in Table 4.3, these are activities and tasks for defining the value proposition and outlining ecosystem structure (Ecosystem View), defining core interaction processes (Interaction View) and the description of necessary services (Service view). The design stage also includes decisions about the underlying technical infrastructure as well as its prototypical implementation and launch (Infrastructure View).

Challenges: Some of the significant challenges in this stage might refer to a missing understanding of the application context. However, modeling decisions based on wrong assumptions about the application context and their specifics todounvollständiger Satz. The considerations from the practitioners' reference models analyzed in Section 1.1, suggests that the best way to start designing an online exchange environment is to focus on one single value proposition that is supported by one core interaction.

4.9.2 Ignition Stage

The priority of the ignition stage is to build the critical mass of participants in order to establish the DMS as a market-oriented environment. On the one hand this network is a network of participants (shaper and enabler roles), and on the other hand, a network, which provides the infrastructure, since each of the DMS participants has a dual role. Given that, *the primary concern is to build a critical mass and clear friction and bottlenecks* in order to ensure a smooth functioning on the operational level.

Related activities: Whereas the previous stage focuses on instantiation and launch of a DMS for an application context, the activities in the ignition phase strive toward improvement and evolvement (see Table 4.3). The relevant activities and tasks are for example reviewing participants' experiences and clearing frictions among them (Ecosystem View), improving core interaction and eliminating bottlenecks in the processes (Interaction View) and enhancing of existing services (Service view). Regarding the technical infrastructure (Infrastructure View), there are activities to remedy deficiencies, as well as improving and scaling-up in order to support a growing network.

Challenges: One of the main challenges of this stage can be to ”miss momentum” after the launch. Thus, to miss momentum to gain the critical mass of participants (consumers and providers) and consequently establish positive network effects. Various strategies exist to cope with this issue and propose different approaches to attract new participants in order to make the market-oriented environment more valuable for further participants (e.g., strategies by [PVC16]). Another challenge related to this stage may be to miss establishing an adequate level of trust in the DMS ecosystem. Therefore, emphasis on this stage needs to be on the implementation of services, which contribute to the DMS being able to guard against misbehavior and outright fraud (cf. Section 4.7.2).

4.9.3 Maturity Stage

The priority of the maturity stage is to preserve the ecosystem and ensure the ecosystem's resistance and health. Resilience and health refer to the ecosystem's capability to face and survive disruptions and continue to be productive in creating value for all participants [IL04a]. Consequently, *the primary concern is to retain the existing network* as the source of value creation *and connect to other networks* to gain new value sources for development and growth.

Related activities: Activities related to this stage refer to altering the ecosystem's structure and value proposition to meet participants' needs that evolved during the ignition stage. This also includes activities and tasks for definition and integration of additional action flows and potentially new roles necessary for supporting an extended value proposition (Ecosystem view). Further on, these are also activities to review and optimize existing processes and associated services, but also to rethink and introduce new ones for the extended value proposition to materialize (Interaction/Service View). Consequently, the changes in the upper ecosystem structure must be supported by the

Table 4.3: Life stages of distributed market spaces, their concerns and activities linked to the views dimension

	Design	Ignition	Maturity
Concern	Blueprint, prototype, and launch.	Build critical mass, clear frictions, and bottlenecks.	Retain existing network and connect to others.
Ecosystem View	Define value proposition and outline ecosystem structure.	Review participants' experiences, eliminate frictions.	Alter and extend ecosystem structure.
Interaction View	Define core interactions and related processes.	Improve core interactions and clear bottlenecks.	Review existing and propose new core interactions.
Service View	Define services to implement core interactions.	Improve and enhance existing services.	Optimize by adding new functionalities and introduce new services.
Infrastructure View	Prototype technical infrastructure and launch.	Remedy deficiencies, improve and scale-up.	Keep infrastructure up-to-date, expand for more service.

underlying infrastructure, which also must be kept up-to-date and expanded for more services (Infrastructure View).

Challenges: Once ignited, and after a critical mass has been reached, the main challenges of the maturity stage can be to protect the achieved position. Defending one's own position includes responding to competitive threats, which may come from ”within” the ecosystem (e.g., envelopment attack) or outside from other ecosystems (e.g., conglomerate or intermodal attacks) [EPV11]. The envelopment attacks usually come from participants that established themselves as influencers. An example may be a provider who uses his influencer position to develop complementary ecosystems and take away the network or at least part of it. Outside threats, on the other hand, point to a sharing of stable networks and building of ”conglomerate ecosystems”. Alternatively, a more radical approach may be to try to eliminate the competitive ecosystems by taking it over, as the case with intermodal attacks. However, mature ecosystems have different options to react and fend such attacks. The business ecosystem health concept [HTV06] and ”5E” approach [RR17] give an overview of strategies to help to deal with such challenges and preserve the ecosystem within the maturity stage.

4.10 Summary

In this chapter, the reference model for distributed market spaces was introduced. Chapter 4 hence presented the second main contribution of this thesis and thus the primary results of phase 2 and phase 3 of the modeling process (Figure 4.24). The proposed reference model is intended to convey an understanding of distributed market spaces, serve as a blueprint for its analysis, design, and implementation in specific application scenarios and help with further development during its lifecycle.

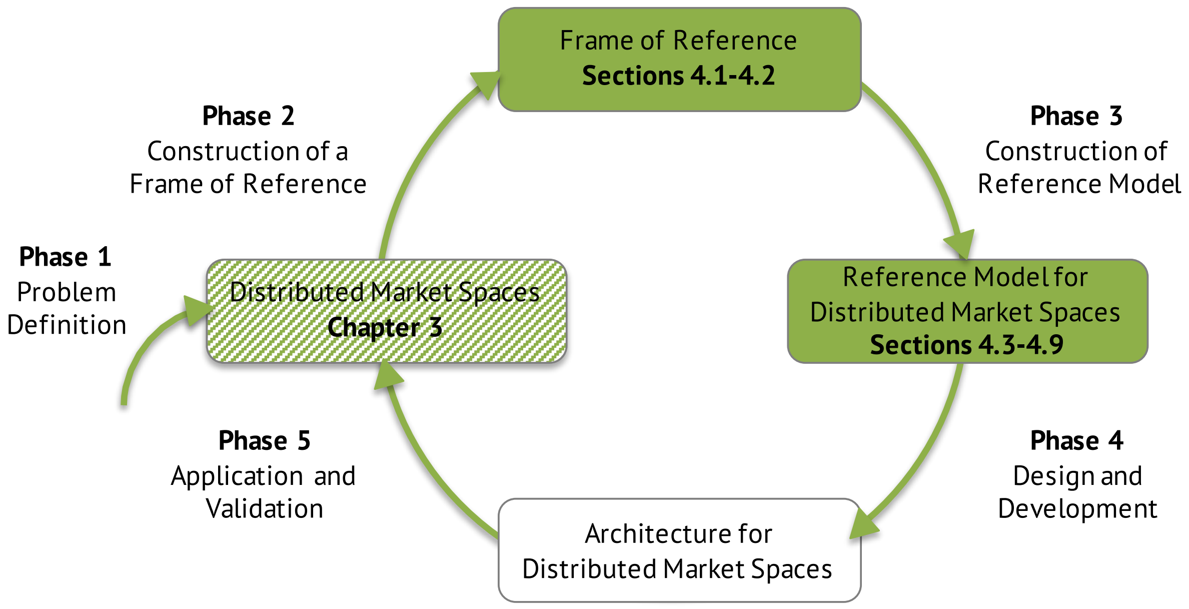

Frame of Reference
Sections 4.1-4.2
Phase 2
Construction of a Frame of Reference
Phase 3
Construction of Reference Model
Phase 1
Problem Definition
Distributed Market Spaces
Chapter 3
Reference Model for Distributed Market Spaces
Sections 4.3-4.9
Phase 5
Application and Validation
Phase 4
Design and Development
Architecture for Distributed Market Spaces

Figure 4.24: Contributions of Chapter 4 to the modeling process

The proposed reference model is a comprehensive multi-view and multi-dimensional reference model that describes the construct of distributed market spaces (reference DMS), taking three different perspectives: phases, views, and stages. A novel phase model of market transactions for complex products is presented to define a reference DMS as a market-oriented environment. It illustrates how to initiate, arrange, and settle complex products to lower transaction costs for complex products. In the following an ecosystem model, an interaction process model, and an underlying service stack are introduced to define a reference DMS on the strategic and operational level. Finally, a life stages model is presented. It acknowledges different concerns and related challenges that distributed market spaces might undergo during different life stages of design, ignition, and maturity.

5 | Architecture for Distributed Market Spaces

This chapter introduces the *architecture for distributed market spaces*. The architecture is designed and developed to serve as a concrete implementation of the infrastructure view of the reference model for distributed market spaces (cf. Section 4.3). Accordingly it represents a blueprint of an information system necessary for the implementation of a reference DMS on the oparational level.

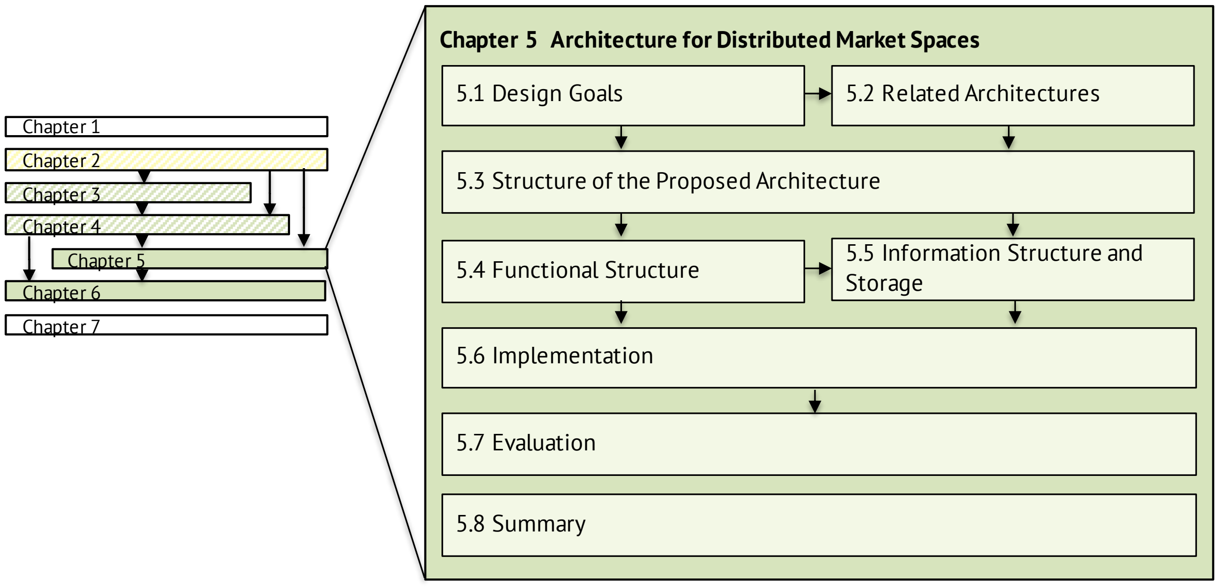

Chapter 1
Chapter 2
Chapter 3
Chapter 4
Chapter 5
Chapter 6
Chapter 7
Chapter 5 Architecture for Distributed Market Spaces
5.1 Design Goals
5.2 Related Architectures
5.3 Structure of the Proposed Architecture
5.4 Functional Structure
5.5 Information Structure and Storage
5.6 Implementation
5.7 Evaluation
5.8 Summary

Figure 5.1: Structure of Chapter 5, its content and dependancies

The primary purpose of the proposed information system architecture is to enable the implementation of foundational, governance, and specialized services as defined by the service stack of a reference DMS (cf. Section 4.7). In doing so, it follows the core principles of the decentralized and self-organized structures on which the reference model is based. According to this principle known as ”form follows function,” the proposed architecture employs distributed resources to perform the required functions, considering the system users as constitutive elements that actively contribute to the establishment of the system structure and its mechanisms. The design of the proposed architecture is represented by its functional structure, information structure, and storage model. The functional structure specifies core components, their responsibilities, and how they interact in order to cover the required functionality and thus realize the required service stack. The information structure defines how information exchanged between functional components is structured and represented. The DMS Ontology specifies the main aspects of the proposed information structure, defining the fundamental concepts, data entities, and primary relationships between them. The information storage model outlines how information is stored and retrieved within the system in a strictly decentralized manner.

Figure 5.1 visualizes the structure of this chapter. Chapter 5 starts with Section 5.1, which formulates the primary design goals, followed by the review of related architectures in Section 5.2. Afterward, Section 5.3 introduces the structure of the architecture by providing a high-level overview of its main building blocks and entities. Section 5.4 presents the functional structure, followed by Section 5.5, introducing the information structure and storage represented by the DMS Ontology and distributed RDF Database model. Section 5.6 presents the prototypical implementation as a proof-of-concept of the proposed architecture. Finally, Section 5.7 evaluates how the proposed architecture meets the design goals for which it is designed, and identifies its main sensitivity points and trade-offs. This chapter closes with a summary in Section 5.8.

The essential parts of this chapter are presented and published in [PRR16], [Hit+16], [RP19c], [RPR20], [RRP21] and [RP19b].

5.1 Design Goals

Design goals on the architecture for distributed market spaces (A_DG_1, A_DG_2, A_DG_3, A_DG_4) are presented in Table 5.1. They are derived from the overall objectives of distributed market spaces as a market-oriented environment (cf. Section 3.3), that is, objectives imposed by the service stack necessary for the implementation of foundational, governance, and specialized services of a reference DMS (cf. Section 4.7).

Table 5.1: Design goals of the architecture for distributed market spaces

Rationale	#	Design Goal	Type
O_1, O_2, O_3	A_DG_1	Functionality	functional requirements
	A_DG_2	Usability	quality property
	A_DG_3	Scalability	quality property
	A_DG_4	Modifiability	quality property

Functionality (A_DG_1) is defined by the functional requirements, that are summarized in Table 5.2. Requirements ($R_1, \ldots, R_8$) describe the most important functional aspects of a reference DMS that the underlying information system must support. This implies that architectural entities and elements must work together in a coordinated manner to support the core activities within the ecosystem, enable interactions among actors (cf. Section 4.2 and Section 4.3), and thus realize the services defined by the service stack (cf. Section 4.7).

Table 5.2: Functional requirements on architecture for distributed market spaces

#	**Functional requirement**	**Description**
R_1	Complex product composition	Empowering consumers of composing complex products as arbitrary combinations of individual products and services.
R_2	Integration of contextual information	Capability of considering the consumers's context, which can encompass much more information than those related to the decision criteria, and include the wider range of constraints and requirements.
R_3	Distributed transactions	Possibility of trading products/services from different providers in a single enclosing transaction from the consumers's point of view.
R_4	Cross-domain transactions	Possibility of trading products/services from different business-domains in a single enclosing transaction.
R_5	Advanced matching	Capability of matching a large number of fragmented, heterogeneous consumers and providers effectively, keeping transaction and coordination costs low.
R_6	Advanced ranking	Supporting consumers in making informed decisions by applying a ranking mechanism that considers context-related constraints to create the "best-fit" list of offers.
R_7	Sophisticated reputation mechanism	Supporting consumers in making informed decisions by considering sophisticated information about potential trading partners.
R_8	Decentralized governance	Enabling decentralized information processing, decision making and data management. Forming an actor-to-actor network and supporting the interactions among actors in a direct and decentralized manner.

Usability (A_DG_2) represents the ability of the information system to support end-users in doing the desired activities efficiently and effectively, which is considered essential to increase user confidence and satisfaction in a self-organized environment without a central authority or control. Usability is usually seen as a quality feature, and as part of the system implementation. But in this case, ease of use is seen as a relevant architectural concern. Usability determines how the system supports the simplicity of use and administration as one of the main design goals of distributed market spaces (cf., Table 3.3, O_3) and thus determines how transaction costs for consumers looking for complex product transactions can be reduced.

Scalability (A_DG_3) denotes the ability of the information system to function over time and to accommodate a large number of actors supporting them to conduct market transactions in many different domains (cf. Table 3.3, O_1,O_2). That requires horizontal and vertical scaling. Horizontal scaling refers to the ability of the system to connect a growing number of different actors with different roles that are essential to solve the problem of critical mass and the associated network effects (see Section 4.9.2). On the other hand, vertical scaling refers to the integration of many business domains, which is essential for seamless cross-domain transactions since complex products might span many domains.

Modifiability (A_DG_4) denotes the ability of the information system to make changes and develop in such a way that the primary concerns of the life stages of a reference DMS (i.e., design, ignition, and maturity) are supported (cf. Section 4.9). This requires that the underlying system be able to make changes to facilitate an incremental development of the ecosystem and thus support scalability and openness (cf. Table 3.3, O_2). This is because initially a ”minimal viable system” is built to run in the ignition phase, and features are added over time in the maturity phase. Therefore, the system’s modifiability is essential for the development of the ecosystem. It allows the integration of new functions and additional services, even when they are not initially planned and developed for this purpose.

The formulated design goals *functionality, usability, scalability, and modifiability* are used as criteria for the evaluation of the architecture for distributed market spaces, as will be shown later in Section 5.7.

5.2 Related Architectures

After the description of design goals, this section reviews a set of existing architectures, concepts, and approaches related to the domain of distributed market spaces. Accordingly, the reviewed solutions for market-oriented environments are grouped around the main categories and assessed in the face of the stated requirements (cf. Table 5.2):

- Electronic Marketplaces
- Decentralized P2P Marketplaces
- Intention Economy (IE)
- Web of Needs (WoN)

Electronic Marketplaces

Electronic marketplaces (e-marketplaces) refer to a market-oriented organization of commercial exchange, aiming to increase market transparency and lower transaction and coordination costs [Bak97]. Established e-marketplaces, also called ”platforms” operate as intermediated and centralized solutions focusing on the technology and transactions of individual products and services. The platform holds a hub position in a network of interactions among different actors [PVC16] and exercises power through centrality and governance considering the terms of access, rules, incentives, and control [HW15]. Regarding the capabilities of supporting complex product composition and integration of contextual information (R_1, R_2), most platforms enable only compositions of products and services within their domain boundaries, i.e., industry, domain, type of products or services, or they offer pre-defined combinations of them, traditionally brought together. An example of a pre-defined complex product is holiday planning that includes booking a flight, hotel, rental car, and guided tour (e.g., offered by Opodo, Expedia, etc.). The contextualization of consumers' requests is predominately reduced to the existing information about products or services. An exception is the 4Caast marketplace [Men+12], an environment for trading cloud services as a modular composition of individual services across different providers. Even though 4Caast is advanced regarding contextualization, it is a domain-specific solution, and as such, it does not support cross-domain transactions. On the other side, well-known platforms, e.g., Amazon Marketplace, Alibaba, eBay, follow the long tail [And06] model and offer a more extensive range of products, especially niche products. By doing so, they basically allow for cross-domain transactions, but only for product and service compositions proposed on their platforms. Electronic marketplaces work well for the transaction of individual products and services, providing reliable support in matching, ranking, and settlement activities for both transaction partners, the consumers, and providers. Yet, they are centralized solutions (R_8) and limited in their ability to support transactions of complex products (R_1, R_3) that need to fulfill a specific consumer-defined context (R_2) and only provide insufficient opportunities to combine transactions of different platforms without switching among them (R_4).

Decentralized P2P Marketplaces

Decentralized P2P (peer-to-peer) marketplaces refer to the concept of a marketplace that brings potential buyers and sellers together to engage with each other directly, without any intermediary [Eym01]. Literature provides different approaches and concepts to shift the prevailing paradigm of well-established marketplaces towards disintermediation and decentralization. For example, concepts by [Eym01],[Sch03],[Ser+08],[XLW10], [HS05], propose different technological solutions, i.e., frameworks and architectures, on how to organize "market without makers" [Eym01]. Yet, they cannot support the whole transaction process in a cross-domain and distributed manner. Some of them are related to a particular product/service domain, e.g., [Ser+08], others are limited on the exchange of services [Kle+17] or even only digital goods [Sch03], and [HS05] supports only certain trading forms e.g., auction-based or supply-oriented trade. Blockchain-based marketplaces represent a group of contemporary attempts to make use of blockchain technology (distributed ledger technology) to shift the whole online exchange onto decentralized environments (e.g., BitMarkets [Pap16], OpenMarket [Ope16], and Open-Baazar [Baz16]) aim to shift the whole trade onto decentralized blockchain-based environments. The main idea of these solutions is to enable providers to create and run online stores in order to sell products and services and connect these stores directly to each other on a global network. This global network connects all online stores following the principles of decentralized governance and enables potential consumers to search for offerings and directly connect to potential transaction partners. As to the literature, [Pap16], [Ope16], and [Baz16] are open-source projects and free of any fees (e.g., [Baz16]), or charge a small transaction or subscription fee (e.g., [Pap16], [Ope16]). Even though all of them support the whole transaction process in a cross-domain and distributed manner (preferably using cryptocurrency), as organization type, they are supply-oriented and focused on the availability of individual products and services. And thus, fall short of supporting market transactions of complex products (R_1, R_2, R_3).

Intention Economy

The Intention Economy (IE), coined by [Sea13], refers to an exchange environment that focuses on a buyers' intention to conduct a transaction with a potential supplier (i.e., vendor) and take control of their relationships with vendors, especially in commercial marketplaces. The main idea of this approach is to switch the current supply orientation towards consumers and their buying needs as the main driver for commercial exchange. IE is represented through the project Vendor Relationship Management (VRM), and its mission is to equip buyers with tools that make them independent in their relationships with vendors and other parties on the supply-side of markets [Ber15]. When using VRM tools, buyers are supported in describing their needs by creating a personal request for proposal (pRFP) and making them visible for vendors in a process called "intent casting". pRFPs are created as persistent computing objects and published in the Kinetix Rule Engine. The process of matching supply and demand is done by many specialized, domain-specific platforms. They support identifying the best and final offers (BAFO), as well as conducting transactions among trading partners. Even though VRM tools and related platforms support personalized requests for proposals, as well as matching with potential vendors, there is no obvious evidence that they support the integration of contextual information (R_2) as well as cross-domain and distributed transactions (R_3, R_4) in a direct manner (R_8).

Web of Needs

Web of Needs (WoN) proposed by [Kle+14] is considered a "need-satisfaction" ecosystem that should serve as a foundation for a distributed and decentralized e-marketplace on top of the Web. The central element of such a marketplace is the 'owner- proxy', an entity anonymously controlled by a user. It contains a description of the task it has been created for. It describes the demand or supply, represents the intention to enter into the transaction, and includes information on the owner. Owner-proxies are published on the Web by sending them to the so-called WoN nodes, which are distributed and interconnected. Published owner-proxies are made aware of each other by independent matching services that compare their descriptions and inform them about possible transaction counterparts. The matching services collect information in a similar way as Web search services by crawling through all WoN nodes. If potential transaction partners are identified, a matching service sends hint messages to both of them. Then, owner applications can initiate a transaction by sending the contact message to the potential transaction partner. While WoN provides support for describing needs, in the case of complex products, the consumer has to publish atomic entities and wait for responses and thus manage the compositions manually. However, if the consumer wants the WoN to process the whole composition, he can publish a complex need, waiting for a matching service capable of interpreting his complex need. Therefore, the effects of adverse selection are still retained, and native support for the processing of complex products remains insufficient (R_1). Regarding conducting transactions between identified partners, the WoN does not go beyond starting the conversation between them. Also, there is no obvious support for cross-domain and distributed transactions (R_3, R_4).

Closing remarks on related architectures

The consideration above has shown that reviewed architectures, concepts, and approaches related to distributed market spaces as market-oriented environments have strengths and limitations. The electronic marketplace model for commercial exchange is mature for individual products and services and thus, fulfilling R_3, R_5, R_6, R_7. However, it is a centralized solution and falls short of supporting complex products (R_8, R_1, R_2, R_4). Other approaches address some of these limitations, but they do not represent a comprehensive solution on their own. Either they provide tools that need to be integrated with other platforms to be fully usable (i.e., IE), or they address one of the requirements with the decentralized P2P marketplaces, whose main focus is on decentralization (R_3, R_8).

Moreover, as with the WoN approach, they do not support all phases of the market transaction since they focus only on matching demand and supply and thus neglect the additional coordination costs for the consumers looking for the settlement of complex products. Given these limitations, reviewed approaches and concepts are not suitable possible implementations of the infrastructure view of a reference DMS (cf. Section 4.8.2). Therefore, a novel architecture is required to meet the stated requirements and serve as an underlying technical infrastructure of a reference DMS.

5.3 Structure of the Proposed Architecture

The architecture for distributed market spaces represents the conceptual structure of an information system that is designed and developed as the concrete implementation of the infrastructure view of a reference DMS. This information system implements the foundational, governance, and specialized services specified in Section 4.7. and thus enables the constitution and running of a distributed markets space on the operational level. The proposed architecture is highly scalable and strictly decentralized. It follows the same design principles as the concept of distributed market spaces as self-organized and decentralized online structures.

According to the well-known software engineering principle "form follows function", the proposed architecture employs distributed resources to implement the required services in a strictly decentralized manner. Therefore, the primary design concern of the information system is to support actors not only as market participants (taking part in a decentralized, i.e., peer-to-peer market and its mechanism) but also as the constitutive parts, which actively contribute to these mechanisms.

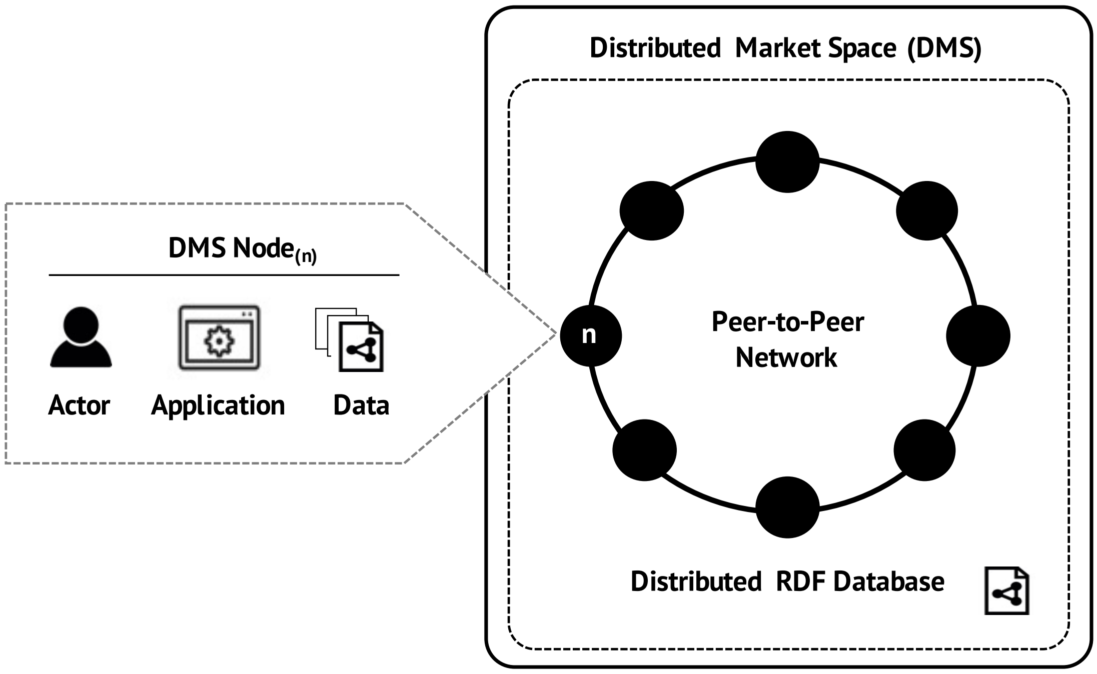


Figure 5.2: Architecture for distributed market spaces

Figure 5.2 shows the structure of the architecture for distributed market spaces and provides a high-level overview of its primary entities:

- Actor
- DMS Node
- Peer-to-Peer Network
- Distributed RDF Database

Actors can be anybody or anything connected to the Internet, defined by the intention to participate within the ecosystem. Actors join the ecosystem by binding to its underlying network, which is organized as a structured *Peer-to-Peer Network*. The primary responsibility of the underlying *Peer-to-Peer Network* is to enable direct communication and interactions among actors and thus implement foundational services (cf. Section 4.7.1).

As a result of these direct connections, each actor makes a two-fold contribution to the system:

- Actors are constitutive parts of the system. They constitute the underlying *peer-to-peer network* and, by doing so, facilitate the ecosystem to build on top of this network.
- Actors are users of the system, and they might take different roles. As such, they contribute to the ecosystem's organizational structure, but at the same time, they satisfy their individual motivation for participation within the ecosystem.

DMS Node is the representation of an actor within the ecosystem. It implements the functionality of governance and specialized services (cf. Section 4.7.2 and Section 4.7.3). These are provided by the user application, which considers the heterogeneity of actors (i.e., human or machine users). Section 5.4 presents the functional structure of the DMS node describing its core components, their responsibilities, and how they interact to support the required functionality.

Distributed RDF Database represents an organized collection of information necessary for the functioning of the ecosystem. On the one hand, this is information relevant for the upholding of the ecosystem's organizational structure, and on the other, domain-related data needed for market transactions in that particular domain. This information is encoded using RDF [MMM+04] and stored on connected *DMS Nodes* by using the operations of the underlying *Peer-to-Peer network*. As a result, each *DMS Node* provides resources for the distributed RDF database, and hence stores a fragment of the global data storage. This provides inherent scalability, as an increasing number of *DMS Nodes* automatically provide more resources in the underlying network. Section 5.5 presents the data model of *Distributed RDF Database* and explains how RDF data is stored in a decentralized manner and made available within the distributed market space.

5.4 Functional Structure

Having overviewed the architecture for distributed market spaces on a high level, the following describes its functional structure. As mentioned above, the underlying information system of a reference DMS is based on the peer-to-peer network, which consists of DMS nodes. These nodes are operated by the involved actors. Figure 5.3 illustrates the conceptual structure of a DMS Node(n).

Actors constitute the system and are at the same time the users of the system. Users might be individuals, companies, and institutions (i.e., human-users) and machines or autonomous agents (i.e., machine-users). Users participate in the ecosystem by using the *application*, which enables them to conduct different activities.

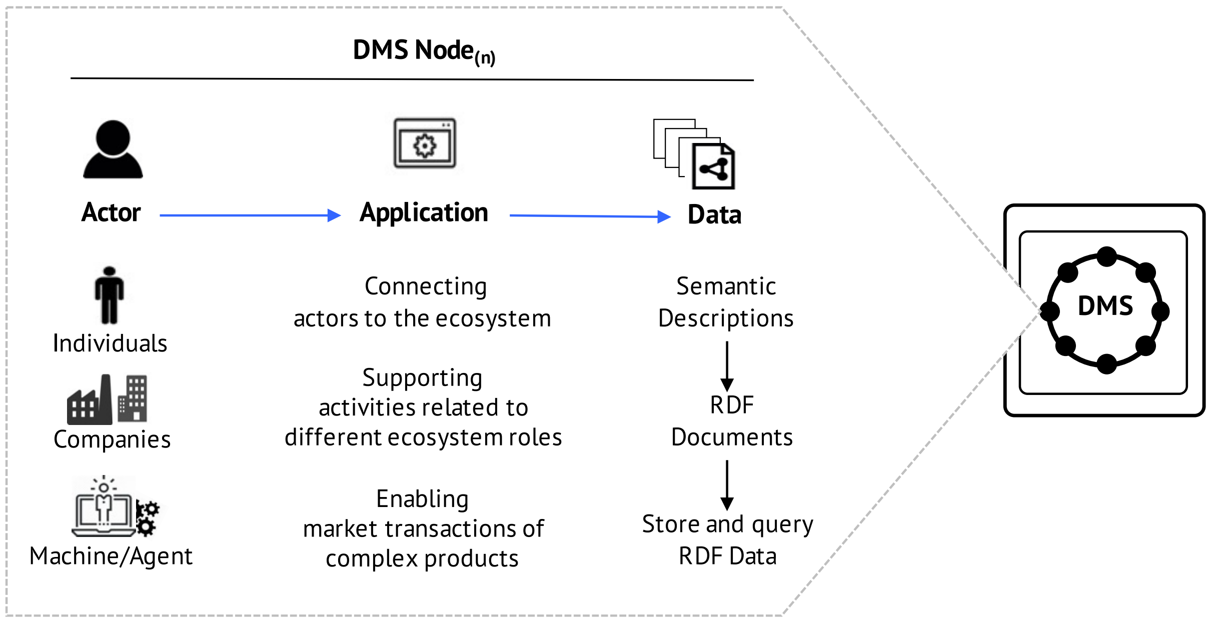

DMS Node(n)
Actor
Application
Data
Individuals
Companies
Machine/Agent
Connecting actors to the ecosystem
Supporting activities related to different ecosystem roles
Enabling market transactions of complex products
Semantic Descriptions
RDF Documents
Store and query RDF Data
DMS

Figure 5.3: Functional structure of a DMS Node

These activities are defined by the interaction phase model (cf. Section 4.6) and specified by the foundational, governance and specialized services (cf. Section 4.7). In summary, they are as follows:

- connecting actors to the ecosystem
- supporting activities related to different ecosystem roles
- enabling market transactions of complex products

For each of these core activities, a DMS node, that is, the *Application* running on this node, provides functions in order to support users conducting interactions within the ecosystem. *Application* consists of functional elements (i.e., core components) responsible for realizing the required functionality. These core components are specified in the following Section 5.7.2 and in Section 5.4.2 which further on specifies interactions among them.

The core components create *Data*, which are further processed, modified, and stored. *Data* created and used are semantically described and encoded as RDF documents. Semantic Web technologies are used because they provide the necessary mechanisms for getting more detailed descriptions of the data involved, as well as incorporate techniques for reasoning on that data. For example, to describe complex products in a domain-agnostic manner, RDF is used as it is the predominant technique for machine-readable representations of knowledge. This allows for the implementation in a complete domain-agnostic way and usage for different application scenarios.

Furthermore, using semantic descriptions enables specifying complex product requests that span products/services combinations from very different domains. Which immediately allows integrating an enormous body of existing ontologies and world knowledge.

For example existing ontologies like GoodRelations [Hep08], BBC core concepts ontology [BBC15], or DBPedia [Ass08]. Section 5.5 specifies the used RDF data model (represented by the DMS Ontology) and describes different RDF documents which are created and modified by functional components and stored in the distributed RDF Database.

5.4.1 Core Components

The core components on which the functional structure of a DMS node is based are presented by the component diagram, as shown in Figure 5.4. It shows the core components, the interfaces they provide and the most important interactions between them that are required to provide the necessary functional capability of a DMS node and thus to meet the specified functional requirements (see Table 5.2). Thereby, the core components are considered distinct functional elements of the system that cover particular functionality and expose interfaces. Interfaces are well-defined mechanisms that allow components to connect to other components and are defined by inputs and outputs, and semantics of each operation they offer [RW11]).

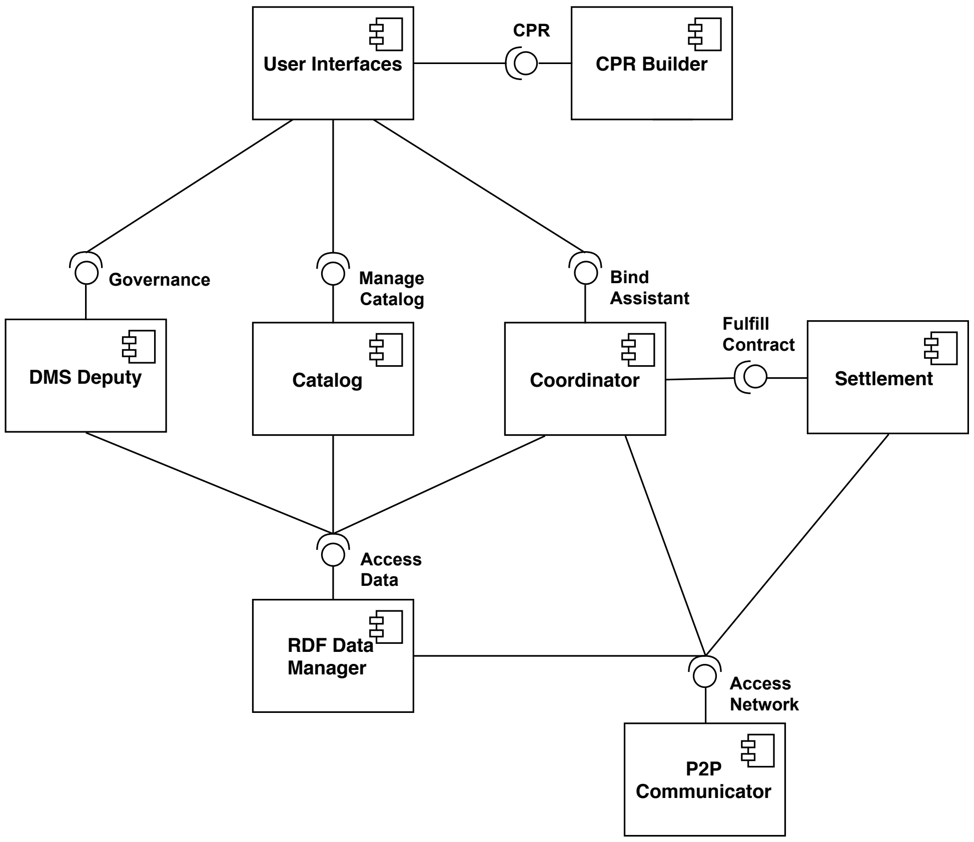


Figure 5.4: Component diagram showing core components, their interfaces and primary interactions among them (UML2.5 [Obj15])

Each of these components has its role, that is, responsibility in delivering the required functionality and thus providing for the functional capability of a DMS node. For example, the User Interfaces component enables different types of users (i.e., humans and machines) to interact with the ecosystem. CPR Builder, Catalog, Coordinator, and Settlement support users conducting market transactions for complex products, and thus cover the functionality of specialized services. The DMS Deputy component enables the ecosystem's governance services and thus allows users to constitute and uphold the ecosystem's institutional life (governance services). RDF Data Manager and P2P Communicator facilitate the components above by providing foundational services for data management and peer-to-peer communication. Table 5.3 summarizes core components according to their primary responsibilities and the interfaces they expose, which will be specified in the following.

Table 5.3: Core components, their responsibilities and exposed interfaces

Component	Responsibility	Interface (exposed)
User Interfaces	Handling of the user's input and application's output considering different types of user-interfaces (i.e., human or machine).	
CPR Builder	Creation of complex product requests.	CPR
DMS Deputy	Enabling users taking an active role in the governance of the DMS ecosystem.	Governance
Catalog	Management of the product catalog within the DMS ecosystem.	ManageCatalog
Coordinator	Managing the process of negotiating complex products, making legally binding agreements and follow-up activities.	BindAssist
Settlement	Management of activities related to the transaction settlement of complex products.	FulfillContract
RDF Data Manager	Storing and acquiring of RDF data.	AccessData
P2P Communicator	Enabling direct communication within the ecosystem.	AccessNetwork

User Interfaces

The *User Interfaces component* is responsible for handling the user's input and application's output considering different types of user interfaces (i.e., human or machine). It enables users to interact with the DMS. On the one hand, this includes the provision of different GUI (Graphical User Interfaces) such as web-based or mobile user interfaces for human users, on the other hand, different APIs (Application Programming Interfaces) and protocols, which enable machines to participate.

CPR Builder

The *CPR Builder component* is responsible for the creation of complex product requests. It leverages the concept of ”genericUIs” [Hit+16] and ”sharableUIs” [HKP17], a novel approach that empowers users to craft personalized user interfaces (UIs) and make them publicly available for the wider community of interest. And by using them to enable users to describe their demands for a specific, complex product.

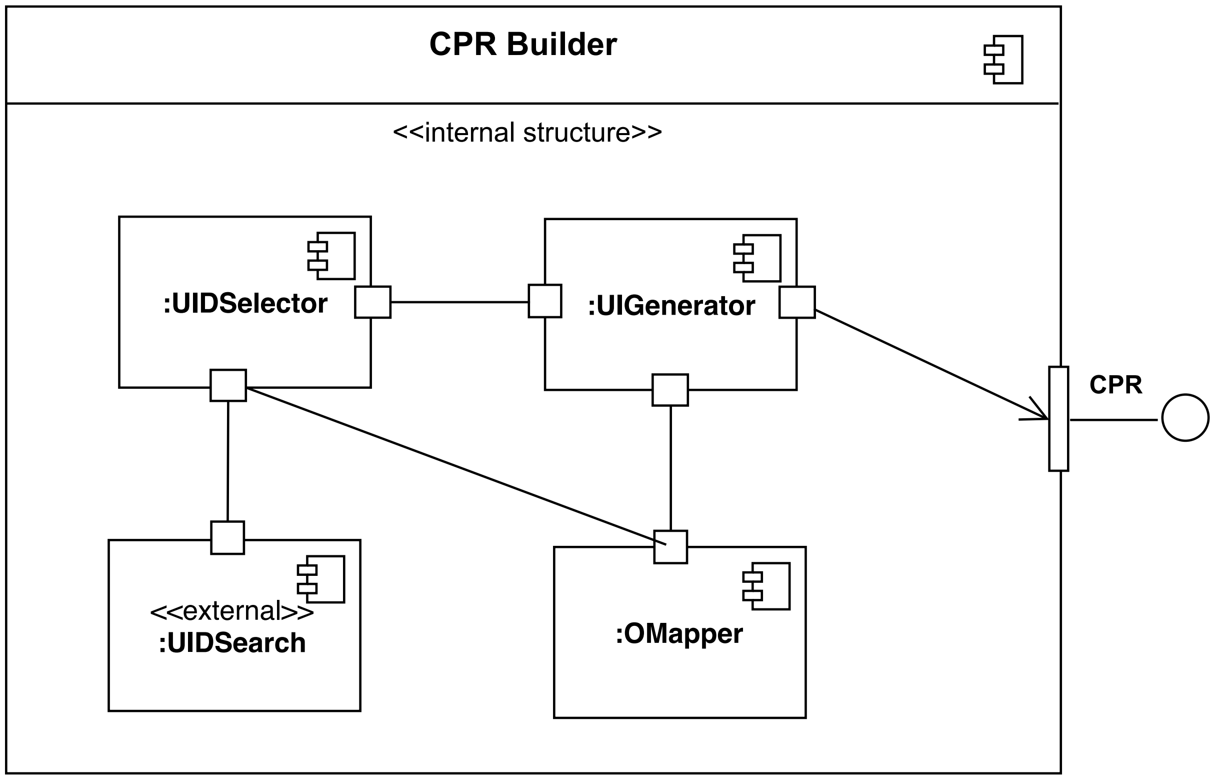


Figure 5.5: CPR Builder component – internal structure (UML2.5 [Obj15])

As presented in Figure 5.5, the *CPR Builder* is composed of four sub-components *UIDSelector*, *UIDSearch*, *UIGenerator* and *OMapper*:

- *UIDSelector* enables aggregating a UI for a specific complex product. It allows users to select suitable, task-related UI descriptions shared as domain-related knowledge within the DMS.
- *UIDSearch* represents a search engine helping users find suitable UI descriptions (i.e., sharableUIs). Sharable UIs relate to a particular domain, and they are usually crafted and shared by DMS members, e.g., members taking the knowledge provider role (cf. Table 4.2).
- *UIGenerator* generates the final UI for the description of a complex product based on the collected UI descriptions. The collected UI descriptions are aggregated into a single UI and presented to the user for data entering.

- *OMapper* maps the entered data for each UI component to instances of the corresponding ontologies. The resulting instances are aggregated into one complex product request (i.e., one RDF document ready for further processing).

CPR Builder component exposes interface *CPR*, that is specified in Figure 5.6.

Interface CPR offers operations for crafting personalized UIs and describing demand for a particular complex product.

Resources

CPR **getCPR**(String memberID)
Parameters

- memberID: DMS user identity

Data Types

- CPR: represents a complex product request. It encompasses a list of separate RDF documents, where each one describes the parts of the demanded product/service combination including related requirements and constraints.

Figure 5.6: Specification of Interface CPR

DMS Deputy

DMS Deputy component is responsible for allowing users to take an active role in the governance of the DMS ecosystem. It is an abstraction of the institutional life and provides transparency in terms of presenting statistics for relevant institutional history events, membership, and member's reputation. The *DMS Deputy component* exposes the interface *Governance*, that is specified in Figure 5.7.

Catalog

Catalog component is responsible for the management of the DMS product/service catalog. It enables creating and registration of new offerings (i.e., product/service descriptions), as well as updating and deleting of existing offerings. It utilizes the afore-mentioned concept of ”genericUIs and sharableUIs” in order to enable the creation of product/service descriptions in a domain-agnostic way. This empowers users to explore among existing UIs and choose one that best suits their offering, to enter product/service descriptions, and generate offerings encoded as RDF documents. Additionally, the *Catalog component* enables the publishing of domain ontologies by storing their URIs as references to domain-related knowledge, which is used for the mapping (see CPRBuilder:OMapper). The *Catalog component* exposes interface *ManageCatalog*, that is specified in Figure 5.8.

Interface Governance offers operations related to becoming a DMS member, checking the reputation of members of interest, and getting various statistics as records of institutional history.

Resources

PMC **getMembershipRequests**(String from, String to, String constraints)
Parameters

- from: request date
- to: request date
- constraints: a search condition

MC **aproveMembershipRequest**(PMC request)
Parameters

- request: request to be approved

RBR **checkReputation**(String memberID)
Parameters

- memberID: DMS member identity

IHD **getStatistics**(Date from, Date to, String constraints)
Parameters

- from: date
- to: date
- constraints: describes required statistics

Data Types

- PMC: represents a pending member card containing a request for the membership that needs to be validated.
- MC: represents a valid member card as the full identifier of each DMS member.
- RBR: reputation bank record encoded as RDF document containing information about members solvency, reputation, and worthiness.
- IHD: represents an institutional history document containing events for the requested period considering further constraints.

Figure 5.7: Specification of Interface Governance

Coordinator

Coordinator component is responsible for managing the process of negotiating complex products, making legally binding agreements, and follow-up activities. It coordinates users' activities and guides them through distinct process steps. It informs about the process's progress and suggests the next activities.

As indicated in Figure 5.9 *Coordinator component* incorporates a set of task-related components *deComposer*, *Matcher*, *OfferEngine*, *RankingEngine*, *Contract* and *FollowUp*. They communicate by exchanging messages coordinated by the *Dispatcher*. Each of these

Interface ManageCatalog offers operations to create and register an offering, update and delete a registered offerings, and for listings the user's offerings registered in the catalog.

Resources

PSD **createProductServiceDescription**(String domain)
Parameters
- domain: a product/service domain

Status **registerProductServiceDescription**(PSD psd)
Parameters
- psd: PSD to be registered.

PSList **getProductServiceList**(String memberID)
Parameters
- memberID: DMS member identity.

PSD **getProductServiceDescription**(String productID)
Parameters
- productID: unique identity of an registered offering

Status **updateProductServiceDescription**(PSD changedPSD, String productID)
Parameters
- changedPSD: changed description of an existing offering
- productID: unique identity of an registered offering

Status **removeProductServiceDescription**(String productID)
Parameters
- productID: identity of an offering in the catalog

Data Types
- PSD: represents a description of an offering encoded as an RDF document
- PSList: represents a list of registered offerings of a DMS member
- Status: represents a structure containing attributes statusCode and statusMessage

Figure 5.8: Specification of Interface ManageCatalog

components is responsible for specific tasks and hence provide specific functions:

- *deComposer* breaks up a complex product request (CPR) into a set of individual product/service requests (PSR). It also receives individual offers and re-combines them into multiple complex product proposals (CPP).
- *Matcher* matches decomposed product/service requests for potential providers and returns a list of potential providers, which contains the information necessary for addressing and sending them RfOs.
- *Offer Engine* enables the creation of product/service offers based on received RfOs and sending offers back to the requesting users.

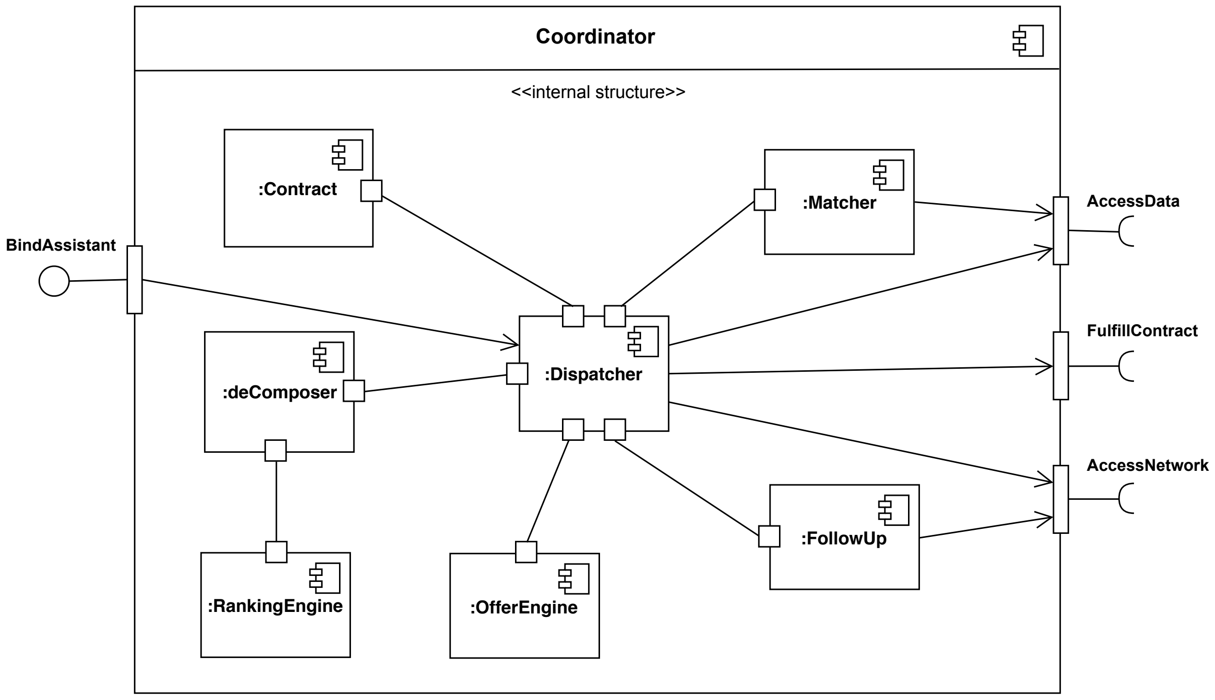


Figure 5.9: Coordinator component – internal structure (UML2.5 [Obj15])

- *Ranking Engine* ranks all received complex product proposals based on the conditions and constraints defined by the user. It generates a list of viable complex product proposals so that the best fit is on top of the list.
- *Contract* manages the two-staged contracting process of an umbrella contract. This includes the creation of a pending contract for each product/service and the creation of a complex product contract which entails all terms and conditions for its settlement and enforcement.
- *FollowUp* supervises the process of after-sales activities and enables the creation of reviews and returns. It enables the initiation of mediation processes and supports decision-making regarding the resolution process.
- *Dispatcher* coordinates exchanging messages among components above as well as over the exposed interfaces (see Figure 5.9). From the set of PSR and the provider list, it creates and sends RfOs to providers. It collects providers' offers and passes them to the deComposer for creating viable proposals (CPP). Based on the rankings of each CPP, it creates a "best-fit" list and presents it to the users. Based on the user's decision it coordinates the two-stage contracting process (cf. Section 4, Figure 4.14).

Coordinator component exposes interface *BindAssisst*, that is specified in Figure 5.10. It uses interfaces *AccessData* (cf. Figure 5.12), *FulfillContract* (cf. Figure 5.11) and *AccessNetwork* (cf. Figure 5.13).

Interface BindAssistant offers operations to start the negotiation and contracting for a complex product request, cancel and review of an order, and to initiate a dispute resolution process.

Resources

Status **questCP**(CPR cpr)
Parameters

- CPR: a complex product request

Status **cancelOrder**(String contractID)
Parameters

- contractID: contract identity

Status **reviewOrder**(String contractID, String review)
Parameters

- contractID: contract identity
- review: a review item

RfM **sendRequestForMediation**(String contractID, String reason)
Parameters

- contractID: contract identity
- reason: a description of mediation request

MR **resolutionAgreement**(String contractID, String decision)
Parameters

- contractID: contractID: contract identity
- decision: recent decision (approval/denial)

Data Types

- RfM: represents a request for mediation encoded as RDF document
- MR: represents a mediation record encoded as an RDF document, containing information about the resolution
- Status: represents a structure containing attributes statusCode and statusMessage agreement.

Figure 5.10: Specification of Interface BindAssistant

Settlement

The *Settlement component* is responsible for the management of a transaction settlement. It assigns identifiers to enclosed transactions, monitors their progress, and takes responsibility for transaction completion, failure recovery, and rollback. It creates a transaction record, which summarizes the most relevant transaction-related data. *Settlement component* exposes interface *FulfillContract*, that is specified in Figure 5.11.

Interface FulfillContract offers operations for settlement of a complex product contract, considering different settlement modes.

Resources

Status **settleContract**(CC contract, String settlementMode)
Parameters

- contract: a contract to be settled
- settlementMode: a way the contract should be settled (trusted/third-party/trustless settlement)

Status **getExecutionStatus**(String contractID)
Parameters

- contractID: contract identity

TR **getTRansactionRecord**(String contractID)
Parameters

- contractID: contract identity

Data Types

- CC: represents a confirmed contract encoded as RDF document. It includes all terms and conditions necessary for the contract execution and enforcement, and it is identified with a unique contractID.
- TR: represents a transaction record encoded as RDF document. It containes all transaction relevant information.
- Status: represents a structure containing attributes statusCode and statusMessage.

Figure 5.11: Specification of Interface FulfillContract

RDF Data Manager

RDF Data Manager provides an abstraction of the DMS data storage handled by the semantic database, which is distributed among DMS nodes. It enables storing and quiring of RDF data within the system. *RDF data Manager* exposes interface *AccessData*, that is specified in Figure 5.12.

P2P Communicator

P2P Communicator provides an abstraction of a P2P overlay network. It enables direct communication within the DMS, integrating discovery and routing mechanisms for finding users of interest and addressing them by sending messages directly and reliably. *P2P Communicator* exposes interface *AccessNetwork*, that is specified in Figure 5.13.

Additional components can be integrated to support roles other than the shaper roles (consumer and provider). For example, these can be components to support other user interfaces for a specific actor type or application context. Or this could be one that contains a Voice User Interface (VUI [Coh+04]), whereby a VUI enables interactions with spoken voice applications (e.g., speech recognition Apps like Alexa [ale] or Siri [inc]).

Interface AccessData offers operations on storing and retrieving RDF data from the distributed RDF store.

Resources

Status **storeRDFDocument**(String document)
Parameters

- document: RDF document to be stored in distributed RDF store.

String **executeQuery**(String query)
Parameters

- document: RDF document to be stored in distributed RDF store

Data Types

- Status: represents a structure containing attributes statusCode and statusMessage.

Figure 5.12: Specification of Interface AccessData

Interface AccessNetwork offers basic operations on the P2P network that enable finding DMS members of interest and addressing them by sending messages.

Resources

put(key, item), get (key), sendMessage(key, message), lookup(item)
Parameters

- key: large binary number representing identifier in the P2P network
- item: data item e.g., a RDF tripple
- message: message to be send

Figure 5.13: Specification of Interface AccessNetwork

And different components to help the creation of ontologies for different domains to support the activities of the knowledge provider role. More specifically, this relates to an OntologyBuilder that is based on a metamodel that contains several patterns used to generate programming control structures to fill ontology instances (for more details, see [RRP21]). However, since the focus here is on the core components to support the users in shaper roles, additional components will be considered part of future work.

5.4.2 Inner-Workings

After describing the core components and their interfaces, the following presents how they work together, interact, and communicate in order to cover the required functionality. The focus lies on shaper roles (consumers and providers) and their activities related to the first three phases of knowledge, intention, and contracting. These three phases are essential for finding and binding potential transaction partners and lay the grounds for the settlement of market transactions for complex products. The relevant processes and specific activities on both the consumer and provider side are defined by the interaction process model (cf. Section 4.9).

Knowledge phase

The knowledge phase includes activities related to getting an overview of the supply and demand within a distributed market space (cf. Section 4.6.1). Providers publish their offerings, and consumers formulate their demands as complex product requests and, based on them, search for potential providers that can provide parts of the required complex product. Figure 5.14, Figure 5.15, and Figure 5.16 presents sequence diagrams (i.e., **sd**s), which illustrate how instances of components *User Interfaces, CPR Builder, Catalog, Coordinator, RDFData Manager,* and *P2P Communicator* realize the functionality of the knowledge phase:

- ***sd*** *Formulating a complex product request*
- ***sd*** *Matching requests for potential providers*
- ***sd*** *Register an offering*

sd *Formulating a complex product request* (Figure 5.14) starts with the user's intention to consume a complex product. To describe the particular complex product, the user first selects a UID (User Interface Description) for each part of the required complex product. UIDSelector:CPRBuilder passes the selected UIDs to UIDGenerator:CPRBuilder which creates the resulting CPUI:CPRBuilder (Complex Product User Interface) and presents it to the user for data to be entered. The user enters requirements and constraints for each part of the complex product and submits the data. CPUI:CPRBuilder sends the submitted data to UIDGenerator:CPRBuilder, and after that, it destructs itself. UID-Generator:CPRBuilder passes submitted data to the OMapper:CPRBuilder. It maps it to the instances of corresponding ontologies and aggregates them into a CPR (complex product request) and sends back the resulting RDF document. This sequence ends with a created CPR representing the starting point for the following processings as shown in Figure 5.15.

sd *Matching requests for potential providers* (Figure 5.15) starts with passing CPR to the Dispatcher:Coordinator which coordinates associated instances of the deComposer:Coordinator, Matcher:Coordinator,:RDFDataManager, and :P2PComminucator. deComposer:Coordinator decomposes the CPR into individual PSRs (product service requests). It sends each of them to the Matcher:Coordinator which forms queries to get registered providers for each PSR and sends them to :RDFDataManager. RDFDataManager passes these queries to the network using the P2PCommunicator. :RDFDataManager catches the incoming data sets and sends them to Matcher:Coordinator. Based on the providers received, Matcher:Coordinator creates an ordered list of potential providers and returns it to the Dispatcher:Coordinator for further proceedings in the intention phase (i.e., sending requests for offerings and getting offers as shown in Figure 5.17).

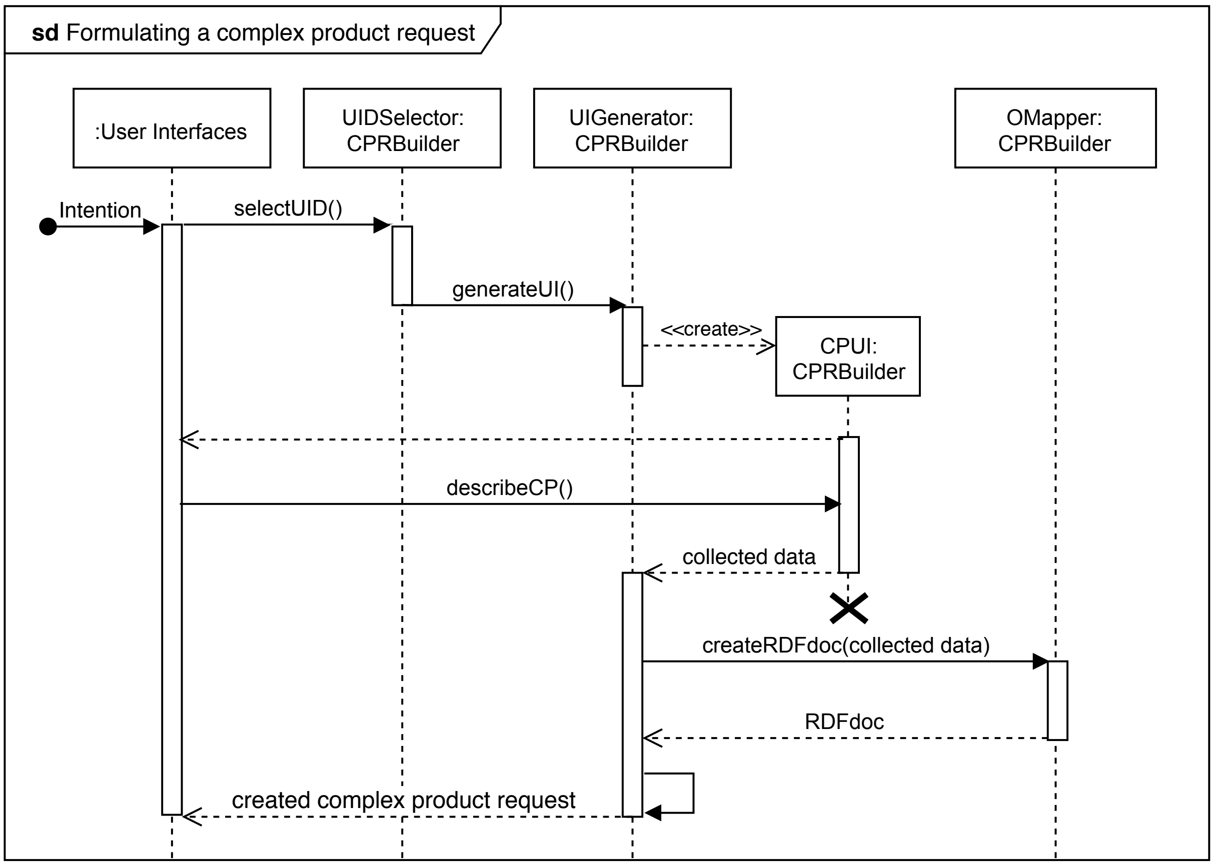


Figure 5.14: Sequence diagram illustrating the communication taking place when the consumer formulates a complex product request (UML 2.5 [Obj15])

***sd** Register an offering* (Figure 5.16) starts with the intention to offer a product or service. For the description of an offering, the user first selects a UID considered suitable for the description of that particular offering. As with the description of complex products, the concept of genericUIs and sharableUIs is also applied here. Accordingly, the Catalog generates the resulting PSUI:Catalog (Product Service User Intercase) and shows it to the user for data entering. The user describes the offering, and after data is collected, Catalog creates a product/service description as an RDF document and passes it to the RDF Data Manager for registration. :RDF Data Manager transforms the resulting RDF document into triples and stores them in the DMS network using the operations of the P2P Communicator. The sequence ends with a notification about the confirmed registration of an offering.

Intention phase

The intention phase for both consumers and providers includes activities related to negotiating conditions for an agreement for the particular complex product (cf. Section 4.6.2). Sequence diagrams presented in Figure 5.17 and Figure 5.18 illustrate how instances of *Coordinator, RDFData Manager*, and *P2P Communicator* realize this

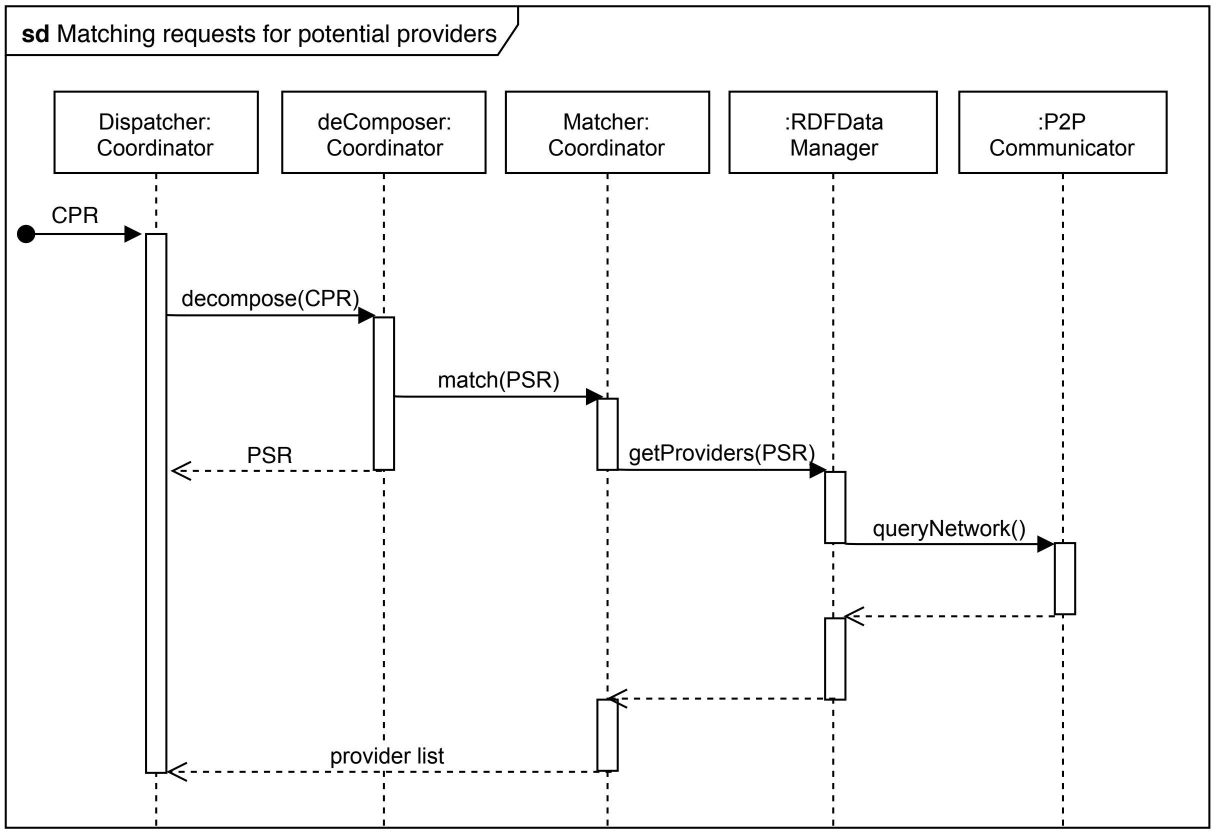


Figure 5.15: Sequence diagram illustrating the communication taking place when matching requests for potential providers (UML 2.5 [Obj15])

functionality:

- ***sd** Sending RfO and getting offers*
- ***sd** Creating viable complex product proposals*

***sd** Sending RfO and getting offers* (Figure 5.17) starts with Dispatcher:Coordinator. It creates requests for offers (RfOs) based on the (decomposed) CPRs and addresses the providers from the provider list. P2PCommunicator then sends the created RfOs to corresponding providers. After receiving a RfO, the P2PCommunicator on the provider side (as indicated by multiple instances) passes these to their Dispatcher:Coordinator. It decides whether to forward the received RfO to the OfferEngine:Coordinator and requires the creation of an offer. In case an offer is created, it will be sent back to the requesting consumer using the same communication path as for receiving a RfO. Dispatcher:Coordinator collects all offers received. The sequence ends after a time specified by the user is reached or after a certain number of offers have been received. The collection of offers received is the input for the following sequence as illustrated in Figure 5.18.

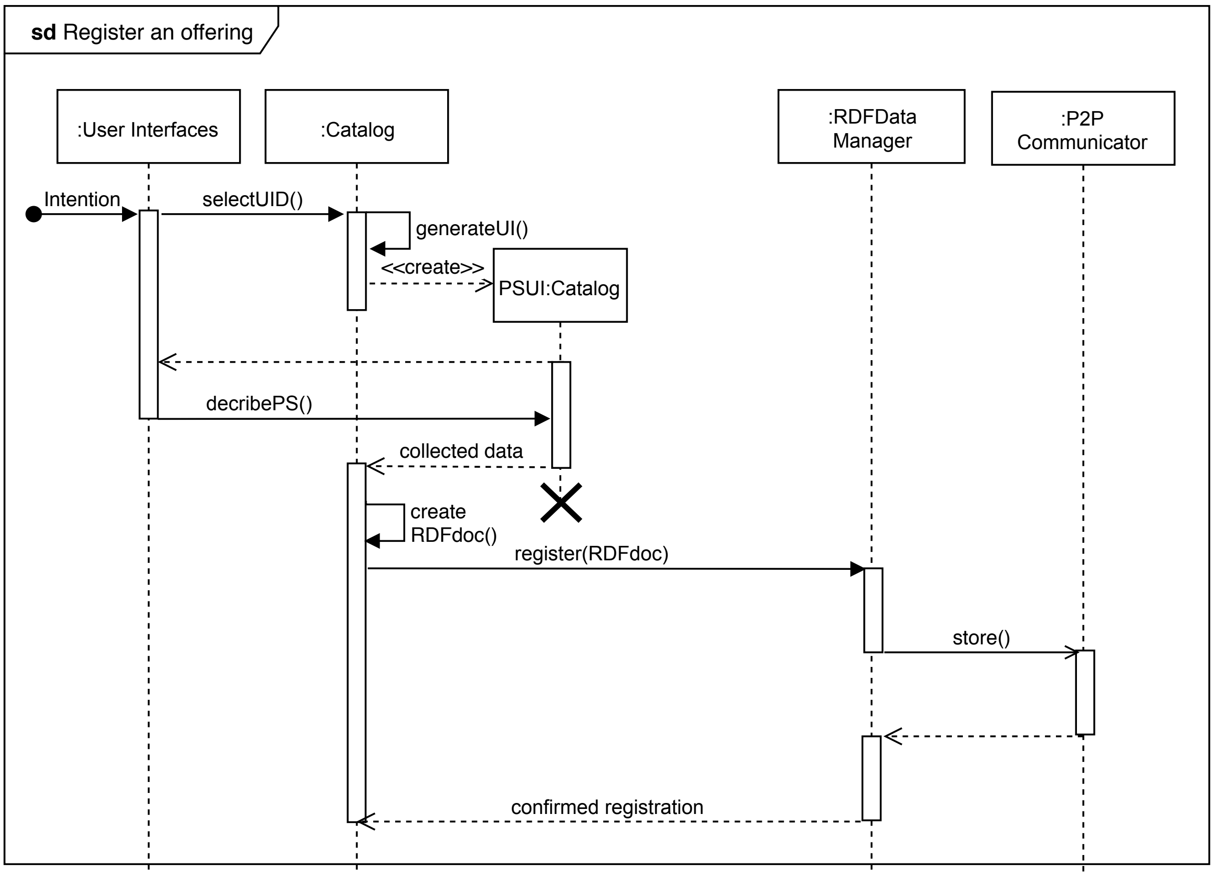


Figure 5.16: Sequence diagram illustrating the communication taking place when a provider registers an offering (UML 2.5 [Obj15])

***sd** Creating viable complex product proposals* as shown in Figure 5.18 begins with de-Composer:Coordinator, which composes complex product proposals (CPP) from received offerings, and RankingEngine:Coordinator that ranks them based on the user-defined constraints and requirements. Dispatcher:Coordinator collects all ranked CPPs:Coordinator, creates a best-fit list and sends it to :UserInterfaces. It presents it to the user for decision-making. The sequence ends with the user's decision. This might favor one specific CPP, or the user decides to go back and redefines the request. In case a particular CPP is chosen, the contracting phase might start.

Contracting phase

The contracting phase (cf. Section 4.6.3) spans the the two-stage process of creating a legally binding contractual agreement between involved transaction counterparts (the consumer and involved providers). The sequence diagram ***sd** Setting up contract* shows how *Coordinator* and *P2P Communicator* instances on both sides manage such an agreement process.

As illustrated in 5.19, this sequence covers the two-stage process of creating a legally binding contractual agreement between involved counterparts (a consumer and providers).

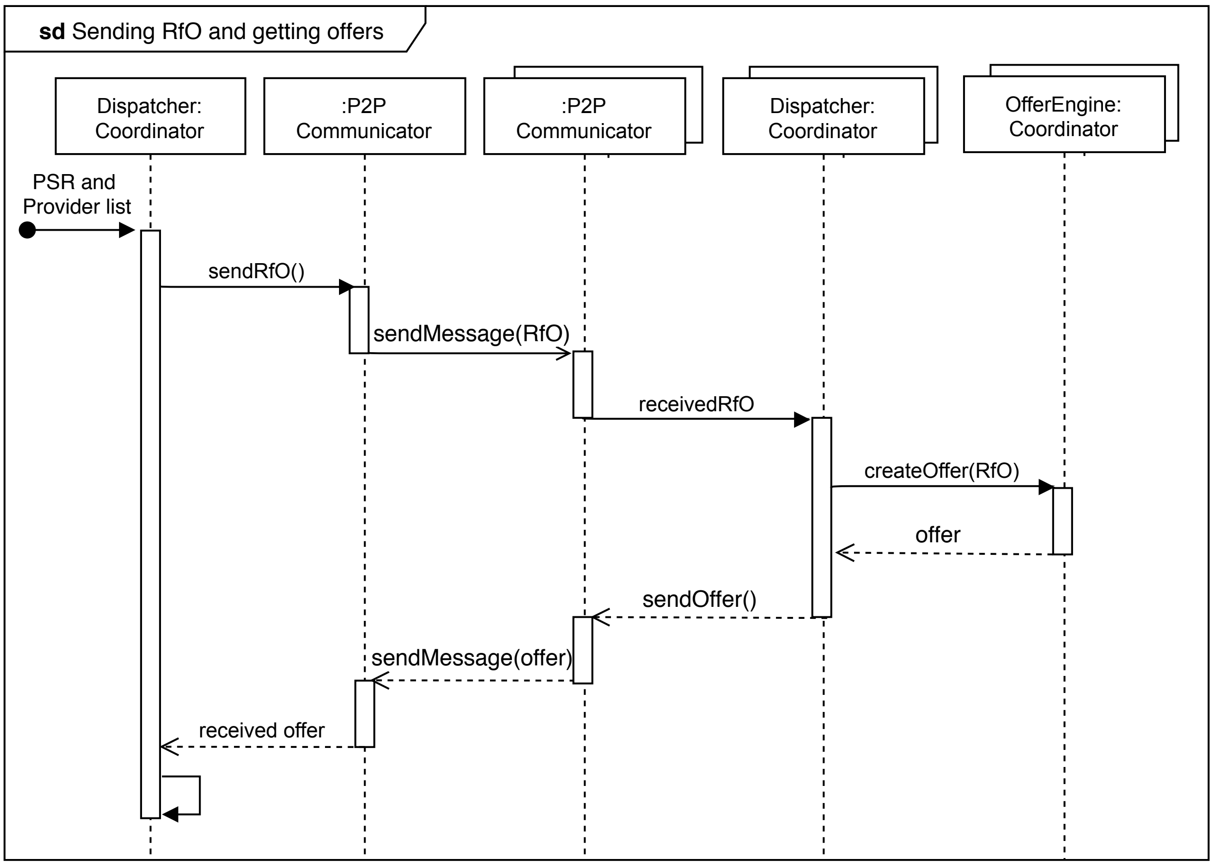


Figure 5.17: Sequence diagram illustrating the communication taking place when sending requests for offerings and receiving offers (UML 2.5 [Obj15])

As with the sequences above, the Dispatcher:Coordinator on the consumer side again manages the distinct process steps. This sequence starts with a message for Contract:Coordinator to create a CPC (complex product contract) based on the agreed CPP (complex product proposal). The created CPC represents a pending umbrella contract, which contains individual pending contracts (PC), one for each product or service in the complex product. P2P Communicator sends each PC to the corresponding counterparts (i.e., providers) for confirmation. The Dispatcher on the provider's side does the confirmation and sends it back to the Dispatcher:Coordinator on the consumer side using the associated P2PCommunicators. After each PCs is confirmed the second stage starts with sending mandatory orders for each of the confirmed PCs. Also, the confirmation process is conducted here, and the consumer's Dispatcher:Coordinator collects all confirmations. The sequence ends with creating the confirmed CPC (complex product contract).

The confirmed CPC is legally-binding and contains all necessary terms and conditions for its execution and, eventually, its enforcement. It is the starting point for the settlement phase where the consumer has to decide which of the three proposed settlement types he wants to choose and proceed with (for different settlement types, see Section 4.6.4). Depending on the consumer's decision, the settlement process differs and even-

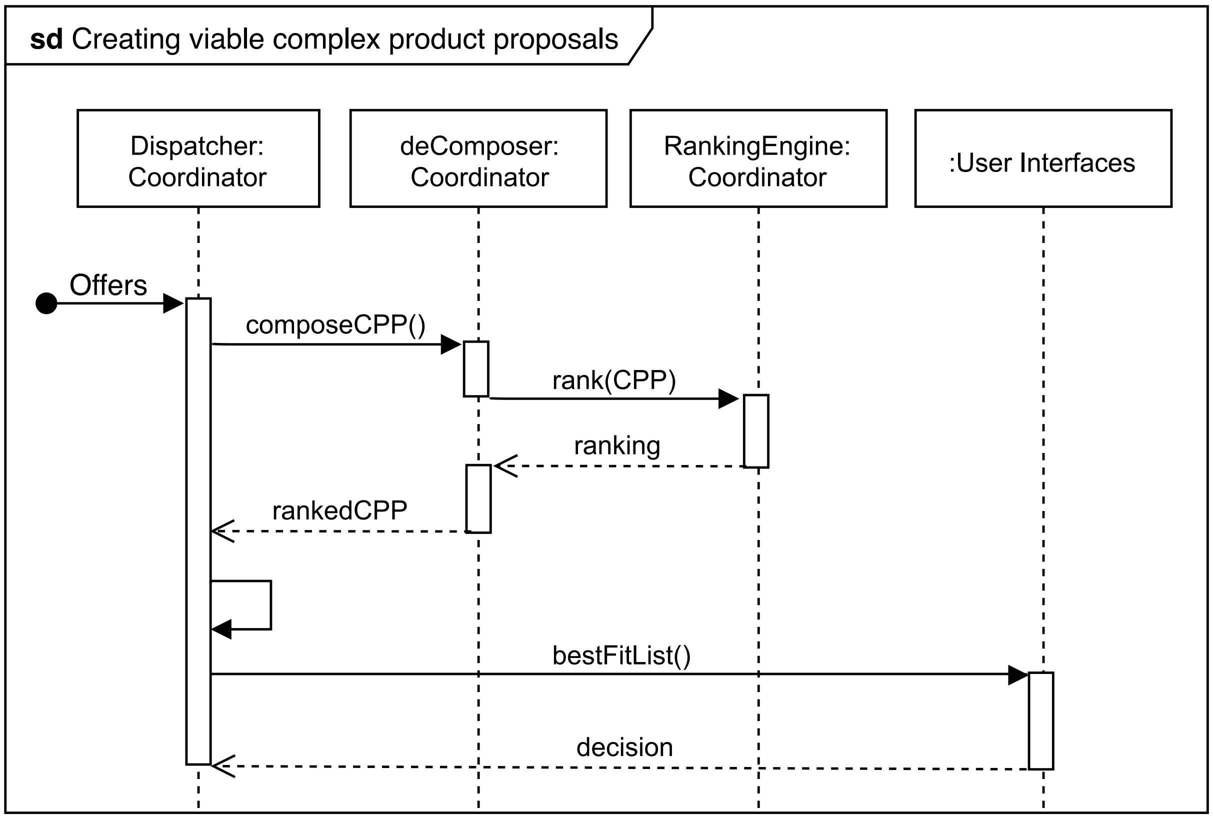


Figure 5.18: Sequence diagram illustrating the communication taking place when creating viable complex product proposals (UML 2.5 [Obj15])

tually requires additional steps, which are considered part of future work and, therefore, outside the scope of this thesis.

5.5 Information Structure and Storage

The primary unit of information exchanged within the aforementioned core components is semantically described data encoded as RDF documents. These RDF documents might have arbitrary content, but their structure needs to follow defined concepts and rules relevant to the particular context for which core components use them. This section focuses on the form and content of information processed, modified, and stored by the core components and thus defines how the DMS system manipulates, manages, stores, and distributes data. The information structure model is represented by the DMS Ontology, determining how data needs to be structured and described in order for the required functionality to be delivered. This is presented in Section 5.5.1. Thereafter Section 5.5.2 concentrates on the storage model defining how data annotated using the DMS Ontology is stored and retrieved within the distributed RDF database.

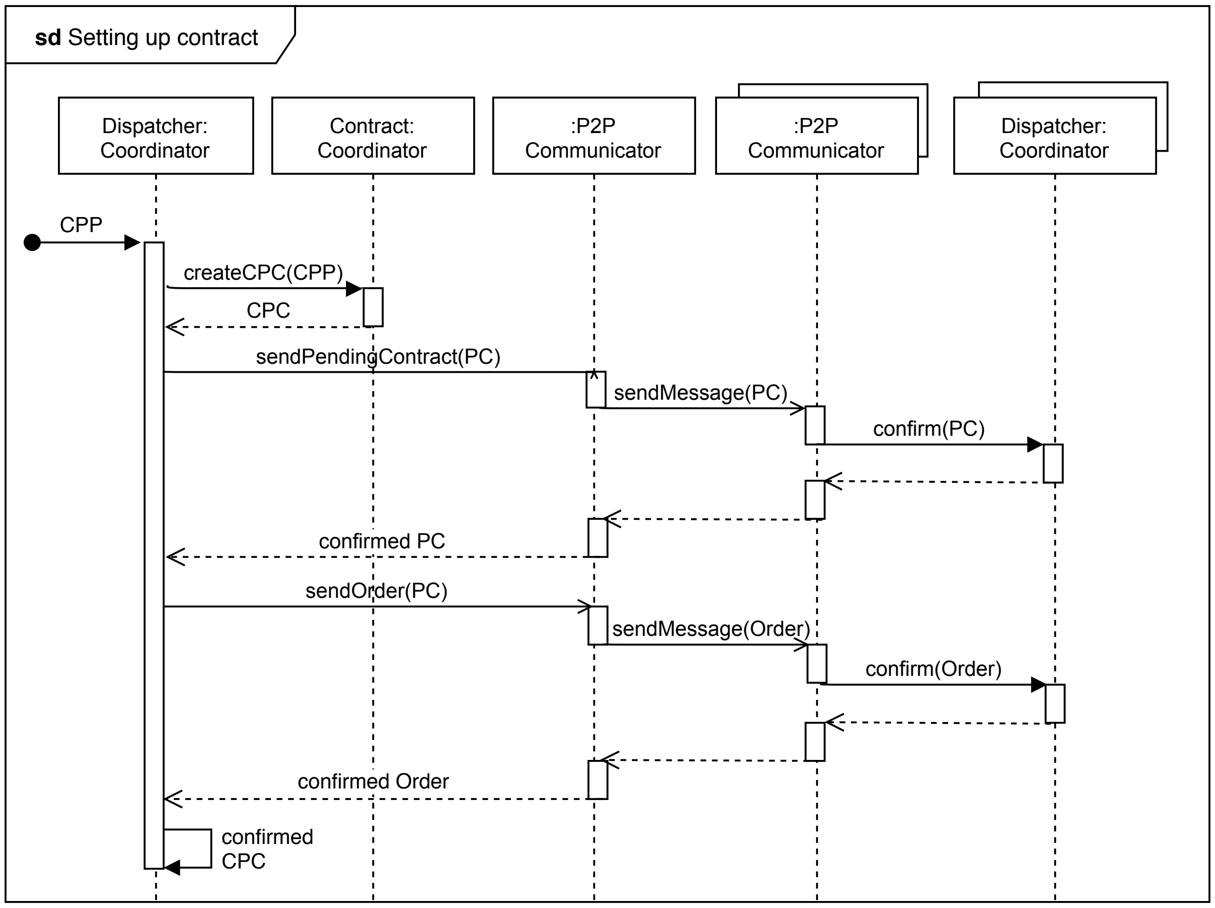


Figure 5.19: Sequence diagram illustrating the communication taking place when creating a legally binding contractual agreement between involved counterparts (UML 2.5 [Obj15])

5.5.1 DMS Ontology

The DMS ontology specifies the main aspects of the information structure on the logical level. It focuses on the static information structure and includes fundamental concepts and their relationships to describe the data necessary for implementing its core functionality, and thus foundational, governance, and specialized services (cf. Section 4.7).

The DMS ontology is modeled using OWL (Web Ontology Language [Deb04]), and it follows the principles of the ”open world assumption”. In contrast to ’closed world assumption’, it "assumes incomplete information about a given state of affairs, i.e., there may be more relevant information than what is provided." [Kee13]. The ”open world assumption” helps to describe knowledge in an extensible way, which is considered essential since distributed market spaces are a new concept. In that vein, the proposed DMS ontology is considered a primary collection of DMS-related knowledge that can be expanded over time to support additional entities and elements that can (and certainly will) result from different application domains.

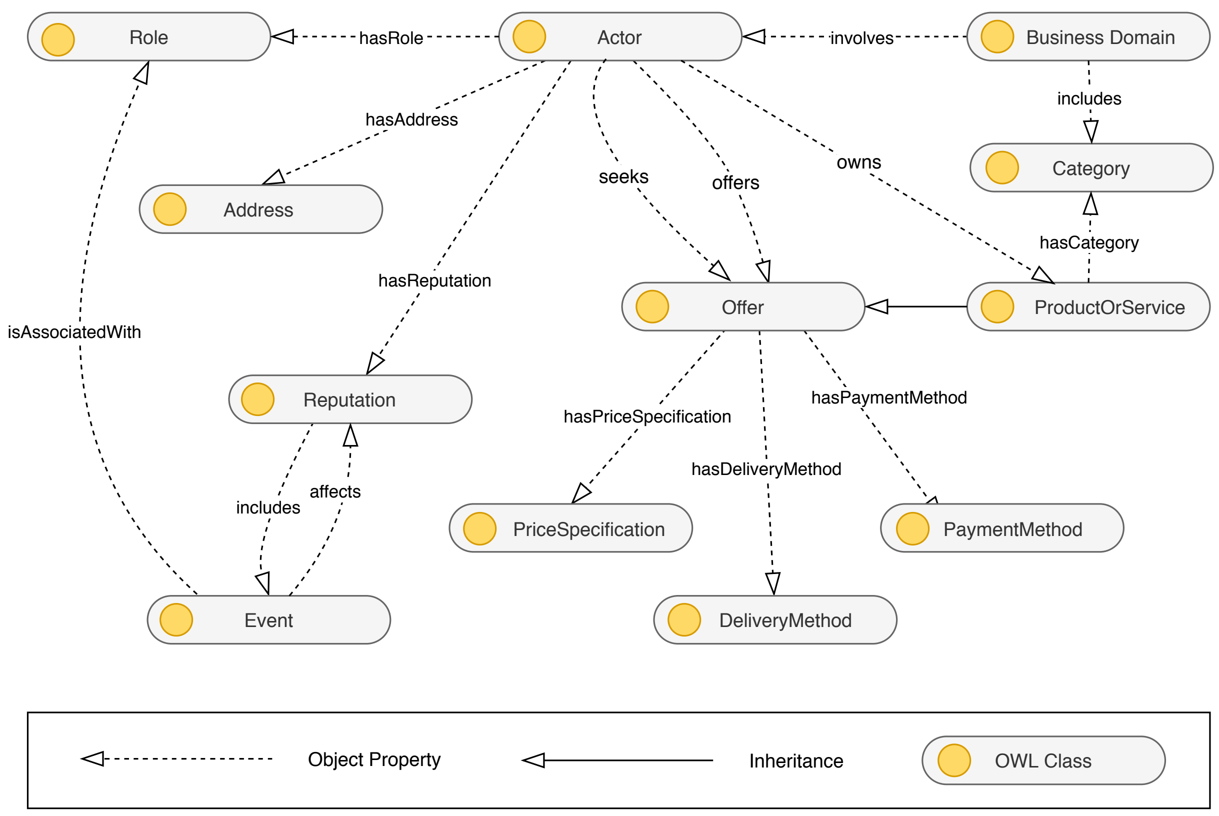


Figure 5.20: DMS ontology

The DMS ontology also leverages the existing ontologies and vocabularies following the best practices in the Semantic Web. As will be explained later, these are ontologies that formalize the knowledge relevant for the implementation of specialized services (Section 4.7.3), in particular existing ontologies for commercial exchange, such as well-known Schema.org [Gro15] and GoodRelations [Hep08].

Furthermore, the DMS ontology extends leveraged ontologies with concepts necessary to integrate specifics of complex products and transactions of such products. It also includes new concepts related to governance activities and, thus, the implementation of governance services (cf. Section 4.7.2).

Figure 5.20 provides a high-level overview of the DMS ontology. It shows its main classes and their primary relationships. These are (at least) the following:

- Actor
- Role
- BusinessDomain
- ProductOrService
- Offer
- Event

- Reputation

Actor class represents an actor who intends to participate in the DMS ecosystem. As previously described, an actor can be anyone or anything connected to the Internet and identified by a global identification number. An *Actor* has a *Role* and an *Address* (*hasRole, has Address*). An instance of *Address class* requires at least the postal address of the actor, and might include other ways of communication with this actor such as, e.g., an e-mail address and/or telephone number. An instance of the *Role class*, on the other hand, describes the role the actor is taking to contribute to the DMS. As previously explained in Section 4.5.2, there are at least six roles an actor may take. For the reasons of simplicity, in this overview, only the roles of shaper (consumer and provider) are considered. An *Actor* taking the role of a provider owns a product or service, which is represented by the *ProductOrService class*.

An instance of the *ProductOrService class* describes a product or service in terms of their specifications, like name and other characteristics of the offered product or service. *ProductOrService* has at least one category which is annotated with (*hasCategory*). *Category class* refers to the corresponding business domain (*Business Domain*), which includes it. Additionally, the *Business Domain class* refers to the involved actor who owns a product or service belonging to this particular domain. When an offer is requested, an actor makes an offering, which is represented by the *Offer class*.

Offer class inherits the emphProductOrService and incorporates additional information such as price specification, delivery, and payment method. As such, the relations of *hasPriceSpecification, hasDeliveryMethod*, and *hasPaymentMethod* concretize an offer in terms of information necessary for the market transaction. Otherwise, when an actor is a consumer, he seeks an *Offer* for a particular product or service.

Event class represents events, that is, something that happens within the institutional life of the DMS ecosystem. This comprises everything from events created by joining the DMS (i.e., creation of member cards) to records regarding settled transactions, reviews, and mediation agreements, as is the case with dispute resolution between actors. *Event class* is associated with *Role class*. That is because specific roles might be involved in distinct sets of activities and thus cause different events.

An instance of *Event class* specifies an event with, for example, event type and description as well as the context. The context considers the event's cause like time and place, whereby place is considered more as the place within a particular process step rather than spatial characteristic. *Event class* relates to the *Reputation class*, as it *affects* the reputation of an actor based on the events related to this actor.

Reputation class represents the statement about the reputation of an actor. Based on the considerations related to the reputation bank in Section 4.5.2, the reputation of an actor is considered a DMS-based reputation which can be computed, for example, through feedback about the actor's transaction histories, but also other statements referring to the actor's contribution to the ecosystem (e.g., "social currency" as explained in Section 4.7.2). An instance of this class expresses a reputation statement, including a reputation value, context, and a time slot in which the reputation value is estimated.

It also might include different aspects of reputation statements, such as role-based or transaction-based reputation.

An exemplary RDF description of an actor annotated using the DMS ontology is presented in Listing 5.1. The resulting RDF document uses the Turtle syntax (Terse RDF Triple Language [W3C14]). The listing starts with the definition of prefixes (line 1-4), which reveals that besides DMS ontology (*@prefix dms:*), rdf and rdfs-syntax, and schema.org are used in this document. Triples that follow state that *Alte Oper* registers as an actor who is involved in the business domain *Ticketing* and that he is accessible through a particular URI (Uniform Resource Identifier) as shown in lines 6–14. *Alte Oper* is identified by *GlobalIdentifier* and *PeerID* (lines 16-19), is registered as a *Provider* and is specified by attributes *Description, LegalName* as well as communication data like email and telephone (lines 20-30). Further triples state details about the postal address as indicated in lines 32-38.

Another example is presented in Listing 5.2. It describes an offering for the annotated provider, as presented in Listing 5.1. As shown, the resulting RDF document uses the same prefixes (lines 1-6) and includes an additional domain-specific ontology for Ticketing *Ticket (TIO)*. It describes a musical event *Chicago The Musical* offered by the *Alte Oper* (line 9), followed by triples stating details about the offer, e.g., name, description, and further price specification (lines 11–25).

5.5.2 Distributed RDF Database

The distributed RDF Database stores information necessary for the DMS to function as a self-organized and decentralized ecosystem. As to the previous considerations in Section 5.3, a distributed RDF Database represents a structured collection of information necessary:

- to uphold the ecosystem and thus information related to membership, institutional history, and reputation-relevant data (as defined by DMS ontology classes Actor, Role, Event, and Reputation)
- to enable market transactions of complex products (as defined by DMS ontology classes BusinessDomain, ProductOrService, and Offer)

Concept and Model

The concept of a distributed RDF Database follows the idea of RDF data storage and retrieval in structured peer-to-peer networks. Such an RDF storage (also known as a peer-to-peer-based RDF store or distributed triple store) represents a relatively new type of a distributed system. It combines the peer-to-peer paradigm with the RDF data model, and by doing so allows a flexible description and exchange of data in large-scale settings [SS06]. Most of the existing distributed RDF store solutions use a structured peer-to-peer system as their underlying infrastructure. As explained in Section 2.2, in structured peer-to-peer systems, data placement is controlled by specific predefined

```
@prefix dms:   <http://dms.org/base/> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix v:     <http://schema.org/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

<http://dms.org/base>
dms:description  "Distributed␣Market␣Space" ;
dms:hasBusinessDomain  <http://dms.org/base/domain/ticketing> .

<http://dms.org/base/domain/ticketing>
a                "business␣domain" ;
rdfs:label       "ticketing" ;
rdfs:comment     "Ticketing␣..." ;
dms:involves    <http://dms.org/base/actor/AlteOper> .

<http://dms.org/base/actor/AlteOper>
dms:hasGlobalIdentifier  "60311-FFM-98797" ;
dms:hasPeerID            "205w270gw8p30udi" ;
dms:hasRole             <http://dms.org/base/actor/role/provider> ;
dms:hasAddress           <http://dms.org/base/actor/AlteOper/address> ;
dms:legalName            "Alte␣Oper␣Frankfurt␣Konzert␣-␣und␣
    Kongresszentrum␣GmbH" ;
rdfs:label               "AlteOper" ;
rdfs:comment             "Die␣Alte␣Oper␣Frankfurt." ;
v:description            "The␣original␣opera␣house␣in␣Frankfurt␣-␣a␣
    concert␣hall␣and␣former␣opera␣house␣in␣Frankfurt␣am␣Main." ;
v:email                  "alte@oper.de" ;
v:telephone              "+49␣63␣1234545" ;
v:web                    "https://www.alteoper.de" .

<http://dms.org/base/actor/role/provider>
rdfs:label               "Provider" .

<http://dms.org/base/actor/AlteOper/address>
v:postalAddress          <http://dms.org/base/actor/AlteOper/
    postalAddress> .

<http://dms.org/base/actor/AlteOper/postalAddress>
v:addressLocality        "Frankfurt␣am␣Main" ;
v:postalCode             "60318" ;
v:streetAddress          "Opernplatz" .
```

Listing 5.1: Exemplary description of an actor using DMS ontology

strategies, i.e., a distributed hash table (DHT) that enables mapping between data and peers, which is considered a good trade-off between scalability and efficiency.

Distributed RDF database enables distributed RDF data management. This means it uses the underlying infrastructure for the distributed storage and management of RDF data among potentially a large number of peers. The data management includes storing, indexing RDF data, and query processing. While the storing of RDF data is influenced by the peer-to-peer network type and its concrete implementation (e.g., Chord [Sto+01],

```
@prefix dms:   <http://dms.org/base/> .
@prefix gr:    <http://purl.org/goodrelations/v1#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tio:   <http://purl.org/tio/ns#> .
@prefix v:     <http://schema.org/> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

<http://dms.org/base/actor/AlteOper> dms:offers <http://dms.com/base/
    offer/ChicagoTheMusical> .

<http://dms.com/base/offer/ChicagoMusical>
a dms:Offer ;
dms:name "CHICAGO␣THE␣MUSICAL"@en ;
v:description "Sinnlich,␣smart␣und␣sexy␣-␣Mit␣Chicago␣kommt␣eines␣der␣
    beliebtesten␣Broadway-Musicals␣in␣der␣englischsprachigen␣
    Originalversion␣in␣die␣Alte␣Oper" ;
dms:includes <http://www.heppnetz.de/ontologies/tio/examples.rdf#
    ticket1> ;
dms:hasPriceSpecification [
a gr:UnitPriceSpecification ;
gr:hasCurrency "EUR"@en ;
gr:hasCurrencyValue "45.00" xsd:float ;
gr:validThrough "2019-07-04T23:59:59" xsd:dateTime ] .

<http://www.heppnetz.de/ontologies/tio/examples.rdf#ticket1>
a tio:TicketPlaceholder ;
rdfs:label "CHICAGO␣THE␣MUSICAL"@en ;
tio:accessTo <http://data.linkedevents.org/event/ChicagoTheMusical> .
```

Listing 5.2: Exemplary description of an offered service using DMS ontology

cf. Section 2.2), RDF data indexing is similar to many existing distributed RDF database solutions. It is based on a hashing of the RDF triples. The query processing is influenced by implemented distribution strategies, which strongly differ among different solutions. For further information on indexing and retrieval in existing distributed RDF databases, see [Fil+10].

Storing and retrieving RDF data among DMS nodes

There are many implementations of distributed RDF data management systems that can be leveraged in the context of the DMS, such as Atlas [Kao+10], RDFCube [MPK06], DecentSPARQL [Mie+13]. Even though these distributed data management systems differ in many ways, they generally support the same data storage and query processing principles. Applied in the case of DMS, these can be summarized as follows. Figure 5.21 illustrates how the distributed RDF base is formed by the contribution of members of the underlying peer-to-peer network. Each member node that is a DMS node has an ID space within the DHT (distributed hash table, cf. Section 2.2), and is responsible for an interval from the value range of a consistent hash function. RDF data are stored on connected DMS nodes in their local RDF database (Figure 5.21). Together all local RDF

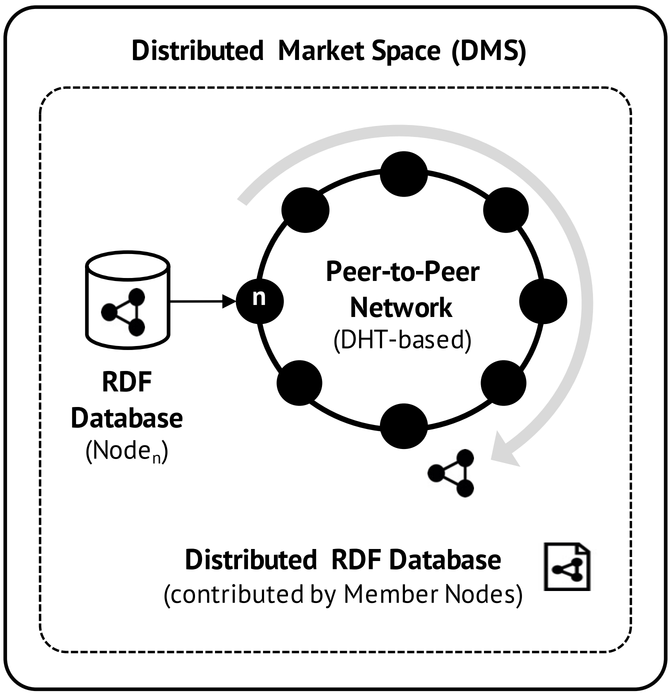


Figure 5.21: Forming of the distributed RDF database

databases build a global database distributed among connected DMS nodes, that is, the distributed RDF database contributed by member nodes.

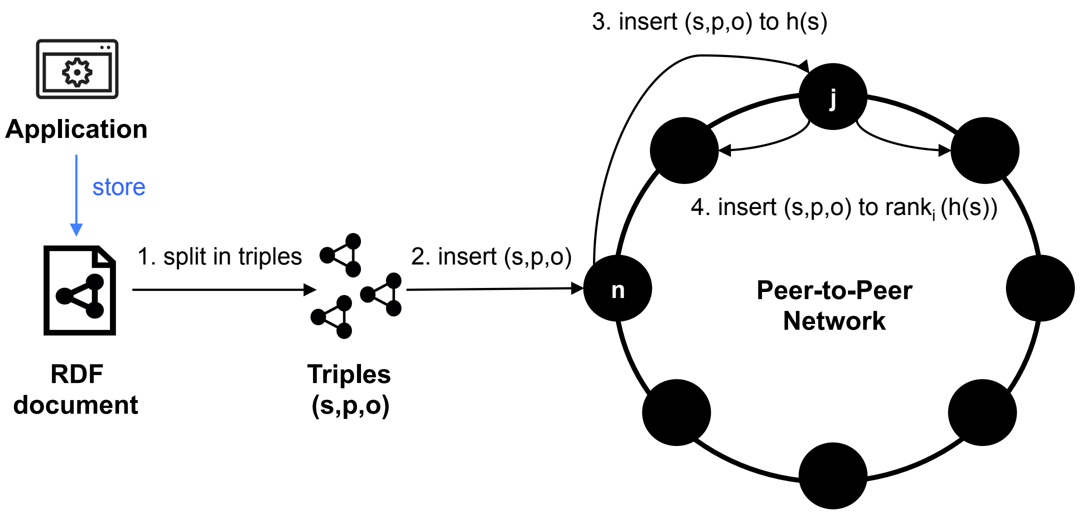


Figure 5.22: The process of storing a RDF Document by a node „n“

When a user application running on the DMS node wants to store data in the distributed database, it needs to submit it in the form of an RDF document. The process of storing this document is illustrated in Figure 5.22. This RDF document is split into a collection of RDF triples (s, p, o) as indicated in step 2. Given the application is running on node *n*, node *n* calculates the hash value of the subject (*s*) of the triple and routes an insert request to the node responsible for this location in the ID space in the DHT (step3). This is node *j*. Node *j* then replicates the triples to nodes in its vicinity, which is illustrated in (step3). The replication is necessary because when the node *j* leaves the network (or it is unavailable for some reason), these neighbor nodes assume the ID space previously covered by *j* and ensure that the triple remains in the network. Steps 2 and 3 are then repeated for the triple's predicate *p* and object *o*.

Alternatively, when the user application requests data from the distributed RDF database, it needs to submit it in the form of a SPARQL query request (SPARQL 1.1 query language for RDF, cf. Section 2.2). As a matter of example, Listing 5.3 shows such a SPARQL query that gets back an ordered list of providers offering services in the domain "ticketing". The listed SPARQL query is related to the sequence diagram (see Figure 5.15) that illustrates the communication taking place between components *Matcher:Coordinator*, *:RDFDataManager* and *:P2PCommunicator*.

```
PREFIX dms:<http://dms.com/marketspace/#>
PREFIX gr:<http://purl.org/goodrelations/v1#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
SELECT ?dname ?sname ?peerid
WHERE {
?domain rdf:type 'domain' .
?domain rdfs:label ?dname .
?domain rdfs:label 'ticketing' .
?domain rdfs:comment ?desc .
?domain bns:hasEntity ?entity .
?entity gr:hasBusinessFunction 'sell'.
?entity rdfs:label ?sname .
?entity gr:BusinessEntity ?be .
?be dms:hasPeerID ?peerid .
}
ORDER BY ?sname
LIMIT 1000
```

Listing 5.3: Ordered list of providers offering in domain "ticketing"

Similar to the storage process shown in Figure 5.22, the SPARQL query is decomposed into a sequence of triple patterns. Then connected nodes cooperate to find RDF triples that match the query's triple patterns. The query processing algorithms and associated strategies for evaluating SPARQL queries can be selected according to the context of the underlying network topology and other factors related to the performance of the chosen network topology. However, these aspects are not considered architectural concerns of the architecture for distributed market spaces as they depend on the respective implementation and are therefore not considered in the design phase.

5.6 Implementation

After presenting the conceptual design of the architecture for distributed market spaces, the focus is on assessing the proposed architectural design. For the assessment, two techniques are combined. First, a proof-of-concept prototype is implemented to validate the assumptions made during the development process and thus to prove that the critical elements of the proposed architecture are feasible. The implemented prototype is then used within a scenario-based analysis. The scenario-based analysis evaluates the overall capability of the system to exhibit its functionality and required properties, and thus to meet the design goals (cf. Table 5.1). Section 5.7 presents the conducted analysis and the final evaluation results.

The proof-of-concept prototype is a software system (i.e., DMS system) that realizes the primary architectural structures of the proposed architecture for distributed market spaces. Still, it contains only a subset of the functionality. The implemented functionality includes functions for setting up a distributed market space as a peer-to-peer based ecosystem, allowing activities among the ecosystem and enabling market transactions of complex products. Accordingly, the prototype implements only a subset of foundational, governance, and specialized services from the DMS service stack (cf. Figure 4.19). These are:

- essential foundational and governance services (i.e., discovery and messaging services, membership service) and
- a set of specialized services necessary to support the primary activities of shaper roles (consumers and providers) within the knowledge, negotiation, and contracting phase (i.e., complex product, product catalog, negotiation, contracting and domain knowledge service).

This simplifies the prototype, but emphasizes the essential aspects for evaluating the conceptual design of the proposed architecture. Additionally, the prototype represents a kind of "skeletal system" that can serve as a test system when embedded in a testbed, which represents the application domain. In this way, it is used for architecture evaluation and as a test system and thus as a starting point for instantiating a distributed market in a specific application environment. Chapter 6 explains this in more detail. In the following, however, the implemented prototype, its building blocks and their concrete implementation are described in detail.

5.6.1 Building Blocks

The prototype is implemented as a Web Application using JavaScript [Jav], HTTP [HTT], and CSS [CSS] as the primary technologies. Low installation costs and flexibility justify the choice of a Javascript-bound prototype. It is simple to set up as it executes directly in a Web Browser, is supported by different devices, and might integrate various relevant interfaces because of various actor types that might engage in a distributed market space.

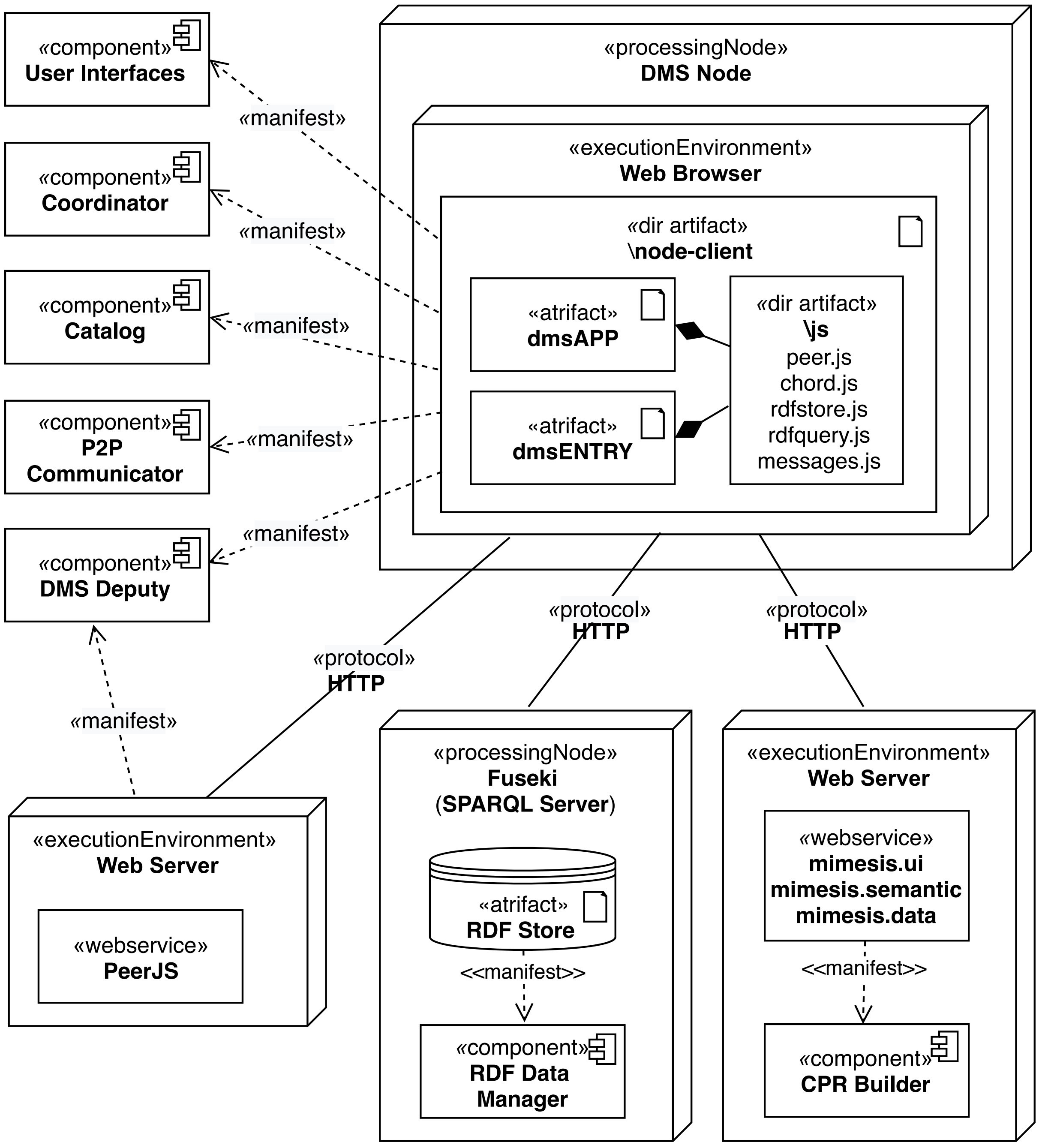


Figure 5.23: Architecture of the proof-of-concept prototype (UML 2.5 [Obj15])

Figure 5.23 presents the deployment architecture of the prototype represented by the *node-client* package. The node-client package runs on a *DMS Node* as the processing node and is executed in a Web Browser as an execution environment. That way a Web Browser represents a DMS node, and all connected Browsers form a peer-to-peer network that underlies the DMS system. Browsers communicate using the WebRTC API [Web], an API for direct exchange of data with Real-Time Communications (RTC) over the Web. The node-client package comprises two main building blocks:

- dmsAPP
- dmsENTRY

dmsAPP artifact implements a user application, which supports users to engage and interact with the DMS. This implementation concentrates on the shaper roles and primary activities in order to exchange complex products and omits the enabler roles. This narrows the scope of the user application, but it shows the implementation of functional components necessary to the handling of complex products. As shown in Figure 5.23, the dmsApp manifests the basic functionality of the components User Interfaces, Coordinator, Catalog, P2P Communicator, and P2P Communicator.

dmsENTRY artifact implements the so-called ”rendezvous point” of the DMS system. The ”rendezvous point’ has two functions. On the one hand, it serves as the entry point for new actors, that is, future users, where they can find information about the DMS and how to join and become a member of the system. On the other hand, it implements the bootstrapping procedure of the underlying peer-to-peer network.

Furthermore, the node-client includes a set of JavaScript libraries (*dir artifact*) used by the dmsApp and dmsENTRY for specific functions. The peer.js [Pee] wraps the WebRTC implementation and simplifies the usage of the included data channel used for data transport between Web Browsers. PeerJS [Pee] is used as a signaling server, as an execution environment for the WebRTC, as it is responsible for generating unique identifiers (peerIDs). The peerIDs are necessary for the bootstrapping procedure and thus the functionality of the DMS Deputy component. PeerJS is integrated as a web service using the HTTP protocol. The chord.js contains functions for peer-to-peer networking, and it implements the principles of Chord [Sto+01] as well as includes the simplified hashing functionality. The rdfstore.js and rdfquery.js [Gar15] represents the implementation of a RDF store with SPARQL1.1[HS]. They are used for the handling of RDF documents in terms of, for example, data processing related to either the decomposition of complex product requests or the composition of viable proposals. Besides these existing libraries, node-client includes further scripts developed to implement diverse helper functions for handling different user activities, and displaying HTML elements to different mechanisms like hashing. The messages.js implements classes that describe the main message types. All messages are JSON [JSO] formated.

The CPR Builder component is implemented as a RESTful Web Service Mimesis [Hit16], [HKP17]. Mimesis integrates the web services mimesis.ui, mimesis.semantic, and mimesis.data. The first two services support the creation of UIs and entering data for a complex product request, and the third creates a complex product request as a resulting RDF document for further processing. This output RDF document is wrapped into a complex product request message. As shown in Listing 5.4, the complex product request message is JSON formated and annotated by fields grouped around two areas.

```
 1 {
 2 from: "<peerID>",
 3 type: "complexProductRequest",
 4 settlementType: "<trusted|trusted-third-party|trustless>,
 5 offerExpectedBy: "<date/time>",
 6 timeToLive: "<date/time>",
 7 payload: {
 8 domain: "ticketing",
 9 [RDF encoded document containing an individual product/service request]
10 }
11 payload: {
12 domain: "parking",
13 [RDF encoded document containing an individual product/service request]
14 }
15 ...
16 }
```

Listing 5.4: Message – complex product request

Lines 2 to 6 contain information about the creator of the request (identified with peerID), the type of the message as well as data that denotes the overall context of the required complex product. These are the ”settlementType”, ”offerExpectedBy” and ”timeToLive” denoting the preferred settlement type (trusted, trusted-third party and trustless as in cf. Section 4.6.4) and date/time by which offers are expected. Lines 7 to 9 (annotated with ”payload”) contain information about the actual request for a particular product or service. It is represented by the field ”domain,” and an RDF encoded document, which semantically describes the individual request. A complex product request message might comprise several payload blocks, depending on the number of products or services embraced by the particular complex product.

For the instantiation of the local RDF database, Apache Jena [Fou17] solutions TDB [Jenb] and Fuseki SPARQL Server [Jena] are used. TDB is initialized as a persistent RDF store (artifact RDF Store) on Apache Jena Fuseki running as a stand-alone SPARQL server. Together they serve as a persistent storage layer and a protocol engine for RDF query and update, running on a processing node. This approach generally emulates the principles of distributed data management and, thus, the main functionality of the RDF Data Manager component. Therefore, further implementations need to replace Apache Jena components with a complete decentralized RDF Database solution. As previously indicated in Section 5.5.2 these might be implementations by e.g., DecentSparql [Mie+13] or Atlas [Kao+10]. However, the implemented DMS data model and the SPARQL queries remain unchanged regardless of the concrete realization.

In addition, source code excerpts are presented that show examples of how certain functions are implemented by the dmsApp. The focus lies on implementing process steps from the sequence diagrams, as detailed in Section 5.4.2, and the concreate functions that realize them.

Listing 5.5 shows the function mathchProvider(). It implements the functionality of matching for potential providers and addressing them for offerings (cf., Figure 5.15). It first queries the SPARQL Endpoint (lines 5-24) in order to get a resulting provider set. Based on the received data, it creates a providerList (a hash map) that holds "peerID" and ”domain” pairs (lines 29-34).

```
1  function matchProvider() {
2
3  var url = "http://h2755784.stratoserver.net:8086/dms-base/query?query
       =";
4
5  var queryProviders =
6  " PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
7  " PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>" +
8  " PREFIX gr: <http://purl.org/goodrelations/v1#>" +
9  " PREFIX bns: <http://business-name-system.org/base/>" +
10 "   SELECT ?dname ?sname ?peerid " +
11 "   WHERE {" +
12 " ?domain rdf:type 'domain' . " +
13 "   ?domain rdfs:label ?dname ." +
14 " ?domain rdfs:comment ?desc ." +
15 " ?domain bns:hasEntity ?entity ." +
16 " ?entity gr:hasBusinessFunction 'sell' ." +
17 " ?entity rdfs:label ?sname ." +
18 " ?entity gr:BusinessEntity ?be ." +
19 " ?be bns:hasPeerID ?peerid ." +
20 "      }";
21
22 url += encodeURIComponent(queryProviders);
23
24 $.ajax({
25 dataType: 'json',
26 url: url,
27 success: function(data) {
28
29 $.each(data.results.bindings, function(i, item){
30
31 var providerRecord = 'domain: "' + item.dname.value + '"  provider: "'
       + item.sname.value + '" with peerID: "' + item.peerid.value +'"';
32 $("#displayProviders").append(providerRecord + "<br/>");
33
34 providersList[ item.peerid.value ] = item.dname.value;
35
36 });
37 };
```

Listing 5.5: Source code excerpt – matching for potential providers

The ”peerID” is necessary for establishing a direct connection with that particular provider and the ”domain” for addressing only these providers with offerings in required domains in order to avoid flooding.

```
...

var count = 0;
for (var providerID in providersList) {

if (!connectedPeers[providerID]) {

var newcon = instancePeerJS.connect(providerID, {
label: 'DMSConnection',
serialization: 'none'
});

newcon.on('open', function() {
connectPeer( this );
});

newcon.on('error', function(err) { alert(err); });

}

count += 1;
...
```

Listing 5.6: Source code excerpt – creating a connection object for addressing matched providers

For each providerID (peerID) from the providerList a new connection object is created and thus a new data channel connection opened as indicated in Listing 5.6. This connection object is then used to directly contact this provider by sending a message of type RfO (request for offer). As shown in function sendRFO(providerID) (Listing 5.7) for each decomposed product/service request, first the domain is checked and in case of a match, a request for offer message is created and eventually sent to the provider (lines 26-36).

```
1  function sendRequestForOffer(providerID) {
2
3  if (myID == providerID)
4  return;
5
6  var conns = instancePeerJS.connections[providerID];
7
8  for (var i = 0, ii = conns.length; i < ii; i += 1) {
9  var conn = conns[i];
10
11 complexProductRequest.forEach( function(psd) {
12
13 if ( providersList[providerID] === psd.getDomain() ) {
14 var msgObject = {};
15 msgObject['typ'] = 'requestForOffer';
16 msgObject['message'] = psd;
17 var msg = JSON.stringify(msgObject);
18 conn.send(msg);
}
});
}
};
```

Listing 5.7: Source code excerpt – addressing potential providers by sending them requests for offer messages

The ”requestForOffer” message (Listing 5.8) contains peerID of the sender (”from”) and of the addressed peerID (”to”) and a date/time on how long this request is valid. Information held in payload includes the specification of a deadline by which an offer is expected to be received, the settlement type and the actual request for the particular product or service encoded as RDF document. That is a part of the decomposed complex product request.

```
{
to: "<peerID>",
from: "<peerID (sender)>",
type: "requestForOffer",
timeToLive: "<date/time>",
payload: {
domain: <domain>,
offerExpectedBy: "<date/time>",
settlementType: "<direct|3rdParty|smartContract>",
[RDF containing the request for offer (decomposed CPR)]
}
}
```

Listing 5.8: Message – request for offer

The corresponding offer message type returned by the offer engine from the addressed provider is shown in Listing 5.9. Offer message type is very similar to the aforementioned ”requestForOffer” message. The difference lies in the payload information related to the validity of the offer (”offerValidThrough”) and the offering itself.

```
{
to: "<peerID>",
from: "<peerID (sender)>",
type: "offer",
timeToLive: "<date/time>",
payload: {
domain: <domain>,
offerValidThrough: "<date/time>",
[RDF encoded containing an offer]
}
}
```

Listing 5.9: Message – offer

All received offers are then re-combined into multiple complete complex product proposals. Listing 5.10 shows the structure of one proposal for a complex product spanning in this example four different domains. The resulting proposals are then passed to the ranking engine incorporated by the Coordinator component.

```
{
to: "<peerID>",
type: "complexProductProposal",
timeToLive: "<date/time>",
payload: {
domain: <domain>,
offerValidThrough: "<date>",
[Payload from offer message]
}
payload: {
domain: <domain>,
offerValidThrough: "<date>",
[Payload from offer message]
}
payload: {
domain: <domain>,
offerValidThrough: "<date>",
[Payload from offer message]
}
payload: {
domain: <domain>,
offerValidThrough: "<date>",
[Payload from offer message]
}
}
```

Listing 5.10: Source code excerpt – complex product proposal

5.6.2 Bootstrapping Procedure

The bootstrapping operation of the DMS system (its underlying peer-to-peer network) is implemented by the *dmsENTRY*. The *dmsENTRY* serves as the bootstrap server (rendezvous server) to which a new or joining user can connect by sending a joining request. For the reasons of simplicity, the rendezvous server is implemented to run on a participating DMS node, which has a known address (fixed URL) and is permanently available and online. The sequence diagram in Figure 6.2.1 illustrates the implemented bootstrapping procedure by the *dmsENTRY*.

When a new user wants to join the DMS, it goes to the entry point and submits a web form with the required input. As a result, a join request message is created and sent to the participating DMS node acting as the bootstrap server. As shown in Listing 5.11, it contains the peerID of the new user, generated by the PeerJS, and a payload containing the data about the potential member, such as name, address, or other information relevant for the creation of the DMS member card. It is sent to the bootstrap server using the WebRTC data channel for bidirectional communications between the browsers, here the new user, and the bootstrap server.

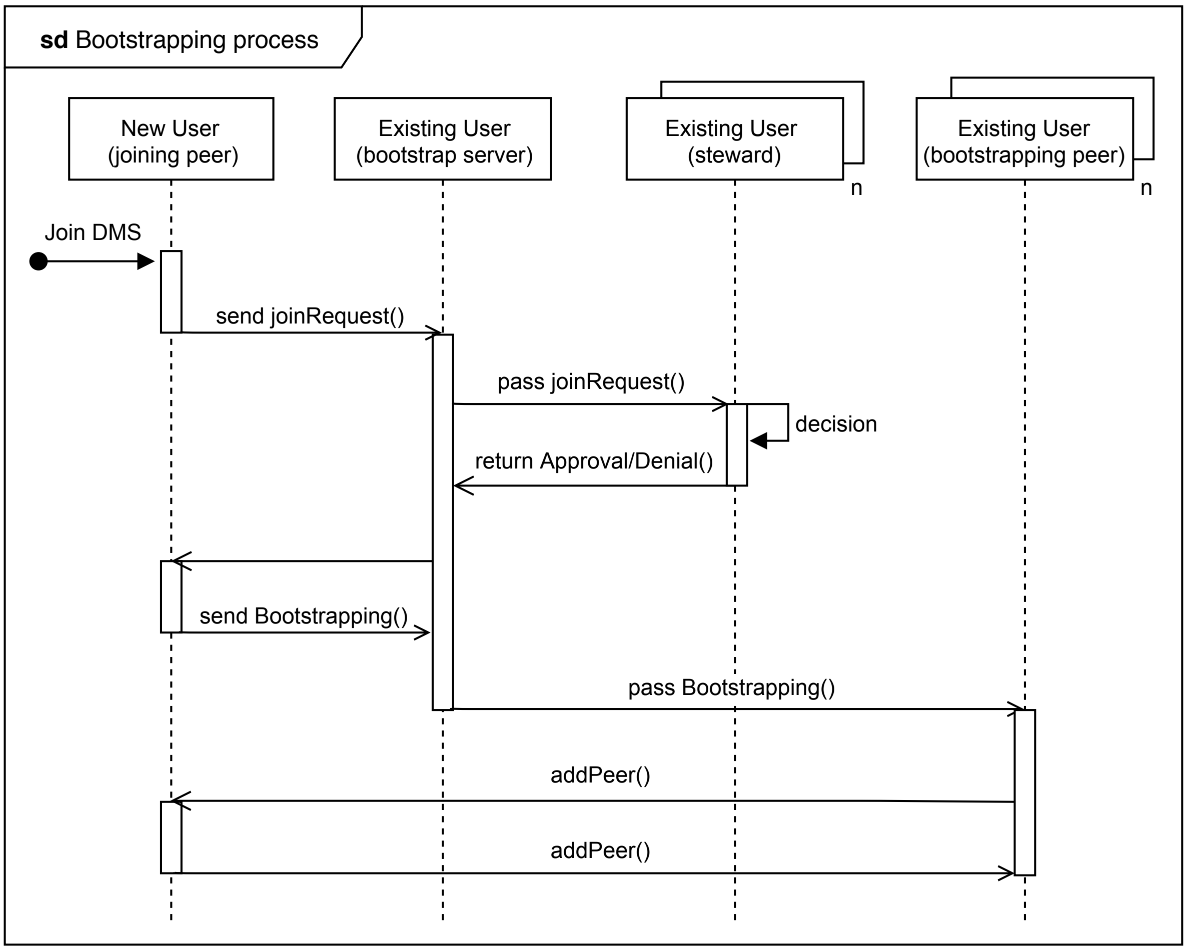


Figure 5.24: Sequence diagram showing the bootstrapping process

```
{
to: "*",
from: "<peerID)>",
type: "joinRequest",
payload: {
type: "potentialMember",
Name: " ",
Address: " ",
...
}
}
```

Listing 5.11: Message – join request message

As soon as the connection object is set up, the bootstrap server selects a candidate peer used for answering joining requests and passes on the joining request message. As previously outlined, these are existing users, i.e., peers, with the dedicated role of a steward, an actor role responsible for answering such requests. However, in this prototype, the existing user accountable for denying or approving DMS membership is the PeerJS server. If the join request is granted, the corresponding answer message is routed back to the bootstrap server, and the bootstrap server routes this to the joining new user.

```
{
to: "peerID",
from: "*",
type: "joinRequestAnswer",
payload: {
type: "memberCard",
memberID: "<memberID> "
}
}
```

Listing 5.12: Message – approval/denial answer from bootstrap server

As shown in Listing 5.12, the answer message contains a memberID relevant for the identification within the DMS. This is because, each DMS user has two IDs: a memberID and peerID. The memberID is considered a permanent identification and relevant for the reputation and self-governance mechanisms of the DMS (see governance services in cf. Section 4.7.2). The peerID serves as the identification in the overlay network and changes as soon as the user disconnects.

The new user must initiate the bootstrapping procedure by sending a bootstrapping message to the bootstrap server, which is then routed to a randomly chosen peer. After receiving the message, the selected peer directly connects to the joining peer. From then on, the bootstrap server is no longer involved. The created connection is used to initiate operations necessary for joining the overlay network. As to the previous considerations in Section 2.2, these are operations related to estimating the joining peer's successors, and starting the stabilization interval to keep the new peer's successor up-to-date and fill the finger table.

```
{
to: "*",
from: "peerID",
type: "bootstrapping",
payload: {
identification: "<memberID>"
}
}
```

Listing 5.13: Message – bootstrapping

If an existing user is disconnected and wants to join the network, only one direct message for bootstrapping is necessary since it already has a memebrID. With the new peerID, it is engaged in sending a bootstrapping message, stabilization and an update of finger tables (last three steps in the sequence diagram above).

Closing remarks on implementation

The bootstrapping procedure implemented by the dmsENTRY applies the so-called mediated bootstrapping as used peer-to-peer systems, like BitTorrent [MS07], and Napster [Coh08]. It implements the bootstrap server using the PeerJS as a signaling server necessary for generating peerIDs, which might be a single point of failure. Recent research offers ways to handle bootstrapping for a WebRTC-based peer-to-peer network, completely removing the need for a central instance. For example, [BKF17] proposes a method based on master peers associated with a DNS-based load balancer, or [Kal00] presents ICR (Internet Relay Chat) for bootstrapping, where a joining peer connects to an IRC channel, where it can find and communicate with other peers. Both approaches might be considered suitable for further development of dmsENTRY toward a fully decentralized bootstrapping procedure.

Furthermore, the description of the prototypical implementation of the architecture for distributed market spaces mainly focuses on the technical aspects and the technologies used to develop the proof-of-concept prototype. The implemented GUIs (graphical user interfaces) of the dmsAPP and the dmsENTRY are shown and explained in more detail in Chapter 6 using the example wemarket.space and a concrete complex product use case scenario.

5.7 Evaluation

This section evaluates how the proposed architecture for distributed market spaces meets its design goals. The prototypical implementation presented above is used as the starting point, from which to evaluate how it covers the functional requirements and meets quality attributes for which it is designed. As previously defined in Table 5.1, these are functionality (A_DG_1), usability (A_DG_2), scalability (A_DG_3) and modifiability (A_DG_4).

5.7.1 Approach and Scope

The evaluation of the architecture for distributed market spaces is conducted using the ATAM (Architecture Trade-Off Analysis Method [CKK+03]). ATAM is a well-known scenario-based architectural evaluation method. The key concept underpinning ATAM is the definition of a set of meaningful usage scenarios, which allow an assessment of the system properties required. Since the required properties are divided into functional and non-functional requirements (i.e., quality attributes), two sets of scenarios are defined:

- *Functional Scenarios (FS)* – to evaluate how well the DMS system will provide its core functionality. These are specific system-usage scenarios for operationalizing stated requirements (R_1,...,R_8).

- *Quality Attribute Scenarios (QAS)* – to assess the ability of the DMS system to exhibit its core quality attributes. These are specific scenarios that illustrate quality attributes to be met.

Defined scenarios have been assessed for its importance (I) and its difficulty of implementation (D) given the presented prototypical implementation (cf. Figure 5.23). Prioritized scenarios that address the most significant quality attributes of the DMS system are refined and summarized into a *quality attribute utility tree*.

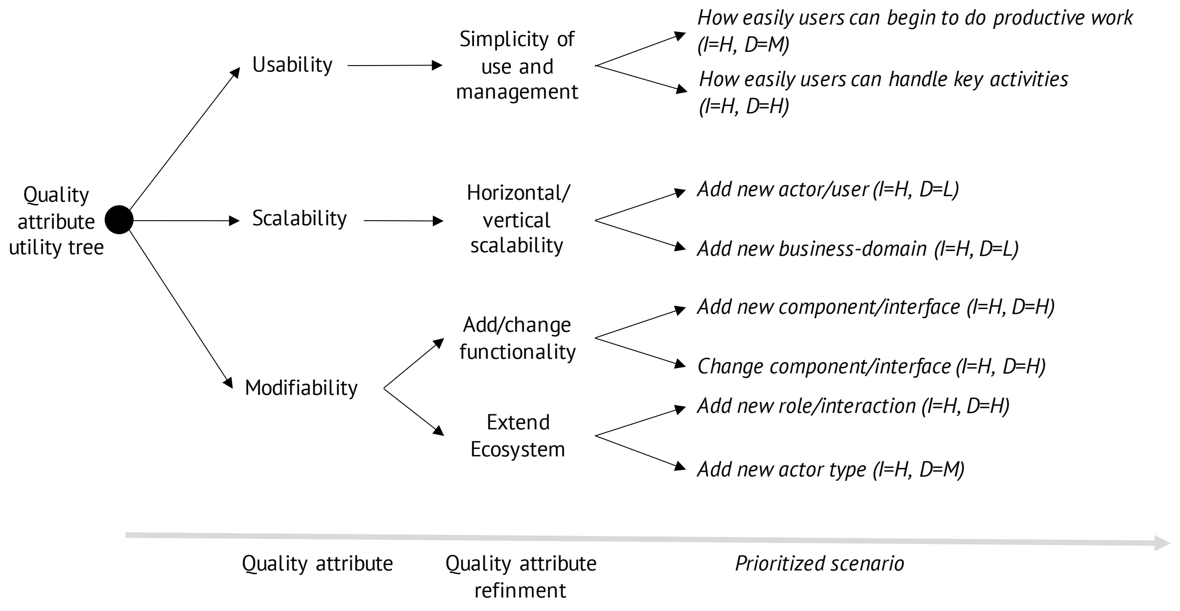

Quality attribute utility tree
Usability
Scalability
Modifiability
Simplicity of use and management
Horizontal/ vertical scalability
Add/change functionality
Extend Ecosystem
How easily users can begin to do productive work (I=H, D=M)
How easily users can handle key activities (I=H, D=H)
Add new actor/user (I=H, D=L)
Add new business-domain (I=H, D=L)
Add new component/interface (I=H, D=H)
Change component/interface (I=H, D=H)
Add new role/interaction (I=H, D=H)
Add new actor type (I=H, D=M)
Quality attribute
Quality attribute refinment
Prioritized scenario

Figure 5.25: Quality atribute utility tree (excerpt)

Figure 5.25 shows an excerpt of the DMS Utility. It presents those scenarios with the highest importance and difficulty, using them as the foundation for the scenario evaluation. Thereby the focus lies on assessing *how the proposed architecture supports those scenarios and which architectural decisions are critical for achieving them*. As a result, the analysis closes with identifying the main sensitivity points and trade-offs of the architecture for distributed market spaces.

5.7.2 Evaluation Results

This section first summarizes the results of the ATAM analysis in the face of the stated design goals. Then it discusses the identified sensitivity points and trade-offs of the architecture for distributed market spaces.

Functionality (A_DG_1)

The evaluation of the functional scenarios indicates that the proposed architecture is feasible as the presented implementation was able to handle them, and thus meets the stated requirements (cf. Table 5.2). As demonstrated, it supports the composition of arbitrary complex products, which span different domains (R_1,R_2,R_4), given existing structured domain-knowledge relevant for commercial exchange in that domain. It facilitates the underlying peer-to-peer network and supports consumers in matching and addressing providers of interest in a direct and decentralized manner (R_5,R_8). However, the prototypical implementation does not cover all functionality needed to support the market transaction process. The distributed transaction (R_3) required to settle parts of a complex product is not yet implemented, as well as the generic ranking algorithms for varying types of consumer-defined ranking constraints (R_6). The proof-of-concept prototype is currently limited to the consumer and provider roles. It does not provide any reputation or feedback mechanisms (R_7) considered the reputation bank's responsibility as the designated role to offer qualified assessments. Therefore, it can be concluded that the primary functionality necessary for the constitution and functioning of the DMS system (see Table 5.1) is feasible and implementable using existing technologies. Functionality (R_3, R_6, and R_7) is partly implemented by other solutions (e.g., [Lim]) and therefore considered an available feature which can be integrated into the prototype.

Usability (A_DG_2)

To be productive in the DMS system, new users need to become a member and join the underlying peer-to-peer network, that is, to undergo the bootstrapping procedure (cf. section 6.1.2). A user only needs to choose in which role he wants to contribute to the ecosystem, and download the client-node package. It includes a user application, which supports tasks related to the selected role or roles. As shown by the implementation (cf. Figure 5.23), the user application can be executed without installation and thus with minimal effort requiring only a Web Browser. The simplicity of use, or how well users can handle key activities, is demonstrated by the user application designed in a way that follows the guided workflow model and system's feedback (see demonstrator in cf. Section7.4). The users are provided with guidance on how to conduct some activities, such as, for example, crafting the personalized user interface for the creation of a specific complex product. Also, users are provided with relevant feedback about the progress of a process monitored by the system. This is demonstrated in the process of sending requests for offers and receiving viable complex product proposals. Such guidance and system feedback minimize the manual effort and cognitive load of users looking for complex products. Hence it lowers coordination and transaction costs, increases use efficiency, and thereby the subjective satisfaction with the DMS system.

Scalability (A_DG_3)

The scalability of the DMS system is obeyed by implementing the peer-to-peer paradigm on the network and application layers. As a peer-to-peer system, the DMS system is inherently scalable, allowing members to immediately join and leave the network while staying stable and continuing to operate. The horizontal scalability increases the available resources of the DMS system and thus lays the ground for more scalability on the vertical scale. The vertical scalability is demonstrated by the opportunity to add a new business domain which enables the DMS system to facilitate market transactions in that domain. This is shown by the functionality that supports the management of domain-related knowledge (e.g., catalog component, see Section). Critical for the vertical scale of the DMS system is, therefore, the implementation of the peer-to-paradigm on the data management level. This is served by the distributed data management approach and the usage of the RDF data model. Distributed data management uses the network capacity of the DMS system to store data essential for the interactions and exchanges among users, for example, matching and institutional history. On the other hand, the RDF data model integrates world knowledge and enables extending the existing and the development of new ontologies (i.e., DMS Ontology), which ultimately facilitates exchange in the different business domains and thus vertical scalability.

Modifiability (A_DG_4)

Modifying existing components and interfaces and adding new ones have modest effects on the change that will propagate among the DMS system. By changing a current component and their interfaces, only the user application on a particular node must be taken offline and restarted. Except for the temporary unavailability of this specific node, there is no impact on the DMS system as a whole. The modifiability of the DMS system is achieved by a high level of encapsulation and separation. The functional structure obeys this, as it consists of loosely coupled core components that communicate via defined interfaces that separate the implementation details from other elements that use them (cf. Figure 5.4). This approach generally simplifies the extensibility of the existing functionality and supports the integration of new ones as long as the defined exchange structure is considered. This also supports scenarios in which a new role and related feature have to be added, potentially new user interactions. In contrast, achieving evolvability of the DMS system has more significant effects as it requires greater propagation of change among the network. For example, consider the scenario in which a new role needs to be added. For users to be able to perform activities related to this new role, new features have to be covered by the user application. Additionally, the interactions between existing roles need to be reviewed and possibly extended to cover the specificities of the new role. Furthermore, the entire system (i.e., each DMS node) must be synchronized. Depending on the number of existing nodes and the extent of the change, the change propagation may take some time, affecting the DMS system's functioning during this period. This should be considered when planning changes in the ecosystem structure, especially when adding new roles or new interaction processes related to different life stages of a reference DMS (cf. Figure 4.23), particularly during the ignition and maturity stage.

Sensitivities and Trade-Offs

The peer-to-peer paradigm applied to the network and the application level of the DMS system assures decentralized information processing and decision making. Still, it bears issues that are inherent to peer-to-peer systems in general. These are well-recognized technical issues ranging from ensuring the availability of service and quality in a strictly decentralized system of autonomous and mostly unreliable peers. Tackling these issues is considered subject to the different implementations of the network topology (cf. Section 2.2) and different bootstrapping mechanisms. An example is the implementation of mediated bootstrapping as applied in the prototypical implementation (cf. Section 5.6.2). Moreover, there are also social issues, e.g., the free-riding problem and selfish peers [AH00], that describe undesired behavior of peers in a strictly decentralized environment. These might be mitigated by applying different trust-building measures and rewarding mechanisms. Examples are peer reputation, and peer- rewarding approaches, such as the concept of a reputation bank (governance service, cf. Section 4.7.2) and social currency [McC14].

The peer-to-peer paradigm applied on the data management level ensures that a large amount of data can be stored using the resources of the underlying network, and thus supports scalability. However, there are two significant challenges inherent in the concept of distributed data management. On the one hand, there are no guarantees that all possible querying results can be retrieved. A possible mitigation strategy is, for example, the application of time-to-live mechanisms defining time intervals in which a response is expected or received results are to be considered. This concept is already utilized in the prototypical implementation and is part of the monitoring function of the user application. On the other hand, the speed of data retrieval (or search speed) within a distributed RDF database is sensitive to the particular implementation. Therefore, data placement, distribution strategies and different strategies of query processing, in particular SPARQL queries, need to be taken into consideration when implementing the system. The DecentSPARQL model [Mie+13] mentioned above allows the implementation of arbitrary data distribution strategies and mechanisms for efficient query processing. Especially it is crucial for complex queries that require data from different peers to be combined.

Another aspect related to data management is the chosen RDF data model considered critical to the description of knowledge and ultimately for the market transactions of complex products. The creation and publishing structured domain knowledge are time-intensive, costly, and require profound domain expertise. Even though this is not an architectural concern, it imposes certain risks and needs to be considered when prototyping and launching the DMS system. That requires a basic set of domain-related knowledge (i.e., ontologies and vocabularies) in the ignition stage. The maturity stage also requires the higher engagement of the dedicated role of knowledge providers to contribute to the knowledge base efficiently and effectively.

Concluding remarks on evaluation

Based on the proof-of-concept prototype and the conducted scenario-based evaluation, it could be shown that the architecture for distributed market spaces meets the design goals to a large extent. The proposed architecture is sensitive to the level of decentralization in relation to the network, application, and data management. It potentially bears an overhead in terms of availability and quality of services, search speed, and no guarantee to get all answers, considered typical for the strictly decentralized environments. Therefore, these aspects need to be reviewed in the context of a particular application scenario and evaluated regarding contextual requirements on the underlying system for that specific implementation.

5.8 Summary

In this chapter, the architecture for distributed market spaces is introduced. Chapter 5 presented the third main contribution of this thesis and thus the results of phase 4 of the modeling process (Figure 5.26). The proposed architecture aims to serve as a concrete implementation of the infrastructure view of the reference model for distributed market spaces. Its primary purpose is to realize the functionality defined by the service stack and fulfill the specific quality properties usability, scalability, and modifiability. The proposed architecture thus represents a software system that enables a distributed market space on the operational level.

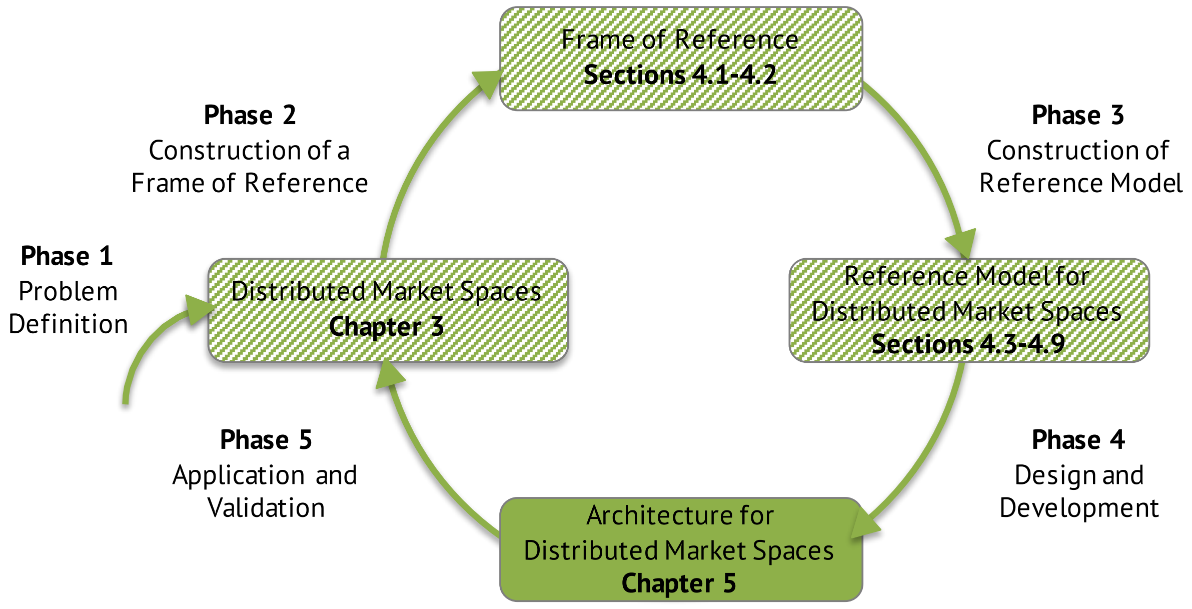

Frame of Reference
Sections 4.1-4.2
Phase 2
Construction of a Frame of Reference
Phase 3
Construction of Reference Model
Phase 1
Problem Definition
Distributed Market Spaces
Chapter 3
Reference Model for Distributed Market Spaces
Sections 4.3-4.9
Phase 5
Application and Validation
Phase 4
Design and Development
Architecture for Distributed Market Spaces
Chapter 5

Figure 5.26: Contributions of Chapter 5 to the modeling process

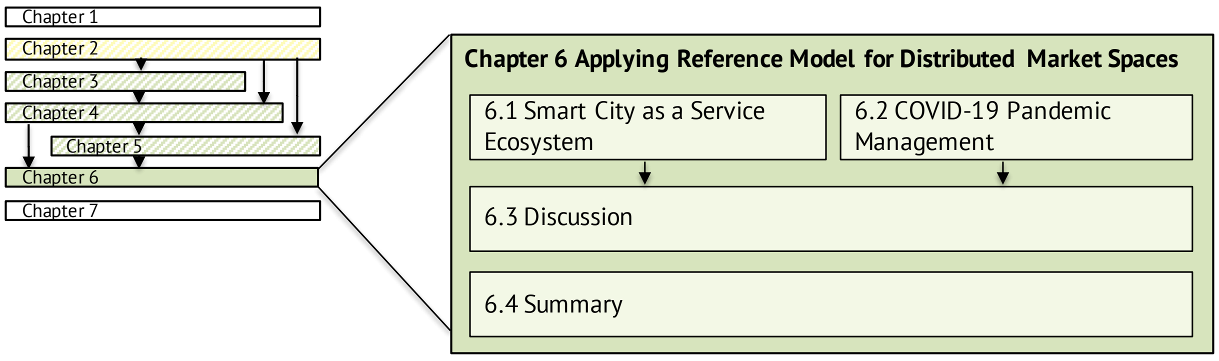
The proposed architecture is described by its functional structure, encompassing seven core components, which realize the functionality of the service stack's foundational, governance, and specialized services. In addition, it defines interfaces and describes behavioral aspects of core components, illustrating how they, when linked together, provide the required functionality. The information structure model, created as the DMS Ontology, introduced the main concepts, entities, and primary relationships necessary for data exchange among core components. The information storage model outlined how information is stored and retrieved within the DMS system following distributed RDF data management principles. The proposed architecture is evaluated by a proof-of-concept implementation and using ATAM as a scenario-based evaluation method. The evaluation results showed that the proposed architecture meets the design goals to a large extent. However, sensitivities and trade-offs are identified, and different mitigation approaches and measures are proposed. Both are to be considered in the context of a particular application scenario.

6 | Applying Reference Model for Distributed Market Spaces

After presenting the *reference model for distributed market spaces* in Chapter 4 and the associated *architecture* in Chapter 5, this chapter illustrates their applicability in two different application scenarios that were chosen in order to illustrate the great variety of potential applications of the proposed reference model. The two applications of Smart City and the COVID-19-VACC platform may seem incomparable at first, but were chosen deliberately. While the model was originally constructed for transaction-oriented markets, the COVID-19-VACC scenario is meant to serve as a case in point illustrating the added value for a platformization of the healthcare sector.

The main aim is to demonstrate how they can serve as a framework for analyzing, designing, and implementing a distributed market space considering different application requirements. Based on that it sets out to evaluate how the proposed reference model meets the design goals for which it was constructed.

The first application scenario addresses the Smart City context and takes up the idea of a Smart City as a service ecosystem represented by the wemarket.space project. The focus is on demonstrating the applicability of the reference model for distributed market spaces and its suitability as a best practice model for designing and implementing an instance of the service ecosystem using the city of Frankfurt as an example.



Chapter 1
Chapter 2
Chapter 3
Chapter 4
Chapter 5
Chapter 6
Chapter 7
Chapter 6 Applying Reference Model for Distributed Market Spaces
6.1 Smart City as a Service Ecosystem
6.2 COVID-19 Pandemic Management
6.3 Discussion
6.4 Summary

Figure 6.1: Structure of Chapter 6, its content and thematic dependencies

The second application scenario addresses the management of the pandemic and the implementation of the National Vaccination Strategy COVID-19 in Germany, represented by the COVID-19-Vacc platform. It is a real-world health case scenario that considers the federal organization of the German state in sixteen federal states and the

European General Data Protection Regulation (EU-GDPR) that imposes strict rules for processing and exchanging personal data. The second application scenario focuses on demonstrating the reusability of the reference model for distributed market spaces. It illustrates how the proposed reference model can be re-used and adapted for a very different domain. Health care is significantly different from the commercial markets for which the reference model was initially constructed.

Figure 6.1 visualizes the structure of this chapter. It begins with Section 6.1. which describes the application context of a Smart City as a Service ecosystem, introduces the wemarket.space project, and presents the results of the framework application. To this end an instance of wemarket.space is launched as a demonstrator in a testbed environment. This testbed was set up as the underlying infrastructure of a wemarket.space to demonstrate the core functionality of the service ecosystem. The following Section 6.2 continues to apply the reference model in the context of the COVID-19 pandemic management. It first describes the contextual requirements and then introduces the blueprint of the COVID-19-Vacc platform as an EU-GDPR-compliant implementation of the COVID-19 vaccination strategy in Germany. Given the instantiated wemarket.space and COVID-19-Vacc, Section 6.3 assesses the applicability and reusability of the reference model for distributed market spaces and evaluates it in the face of the stated design goals. Chapter 6 closes with a summary in Section 6.4.

The essential parts of this chapter are presented and published in [RP19b] and [Rad+21].

6.1 Smart City as a Service Ecosystem

The promise of smart city projects is mainly to facilitate everyday city life by providing useful services that can be consumed by its citizens [RP19c]. Services themselves derive information from data usually gathered from either IT systems or sensors. Therefore, many smart city projects, according to [CDN13], focus on the role of information and communication technology (ICT) as the foundation that allows for the creation of a smart city. However, as cities are complex social systems [KPS13], they cannot be reduced to their underlying ICT infrastructure. As to [CDN13], it is sensible to call a city *smart* when “investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic growth and high quality of life, with a wise management of natural resources, through participatory government” [CDN13]. This definition goes beyond the ICT-centric definitions and covers various aspects of economic and environmental domains, but the definition is vague in terms of ”fueling” a high quality of life, or economic growth. Authors in [AF14] propose a comprehensive definition by viewing a smart city as “an ICT-based infrastructure and services environment that enhance a city’s intelligence, quality of life and other attributes like, e.g., environment, entrepreneurship, culture or transportation”. This work argues that the definition by [AF14] might be extended in a way to embrace value communities and ecosystems as additional aspects of a "city’s smartness."

Definition 1.3: *Smart City* is defined as an ICT-based service ecosystem that enhances a city's intelligence, quality of life and adds value to its participants by facilitating seamless consumer experiences.

This definition does not focus only on citizens, but participants and other municipal stakeholders, giving participants the choice of engaging in a city's ecosystem while taking different roles. Furthermore, this also includes visitors and tourists, as well as businesses offering higher-level commercial services. Such higher-level services can be arbitrarily combined, hence synergistically adding value to the consumers looking for such services. This view of smart cities as service environments is in line with Cisco's definition of the Internet of Everything [Cis13] that "brings together people, process, data, and things to make networked connections more relevant and valuable".

6.1.1 Application Scenario

After introducing the Smart City as a service ecosystem, this section explains the application scenario represented by the wemarket.space project [1]. wemarket.space is a demonstration project that addresses the post-platform scenarios described above (cf. Section 3.2) by showing how Smart Cities can self-organize and form a service ecosystem in a decentralized manner.

The following provides an overview of the wemarket.space project regarding its goal, scope and underlying assumptions.

Goal of the application scenario: The main goal of the wemarket.space project is to facilitate a city's service ecosystem that is formed by active participants who connect to build a value ecosystem for the city. An ecosystem that enables participants to engage in satisfying their demands and, at the same time, to add value to other participants. Hence a service ecosystem that emphasizes seamless experiences for its participants, rather than being a hub for single platforms as currently is the case with, e.g., city's platforms for mobility, transportation, or city events.

Scope of the application scenario: The scope of the wemarket.space project is determined by the design and initialization of a wemarket.space. Therefore, it includes activities related to designing, prototyping, and launching an instance of wemarket.space for the specified requirements and thus for the assumptions defined below.

Assumptions of application scneario: Main assumptions underlying the wemarket.space project are the following:

- wemarket.space is market-oriented exchange environment and supports transactions of individual services but also personalized combinations of them, the so-called city essentials. Hence a wemarket.space enables market transactions of city essentials focusing on the personalized service combinations and thus addresses complex product scenarios as indicated in cf. Section 3.

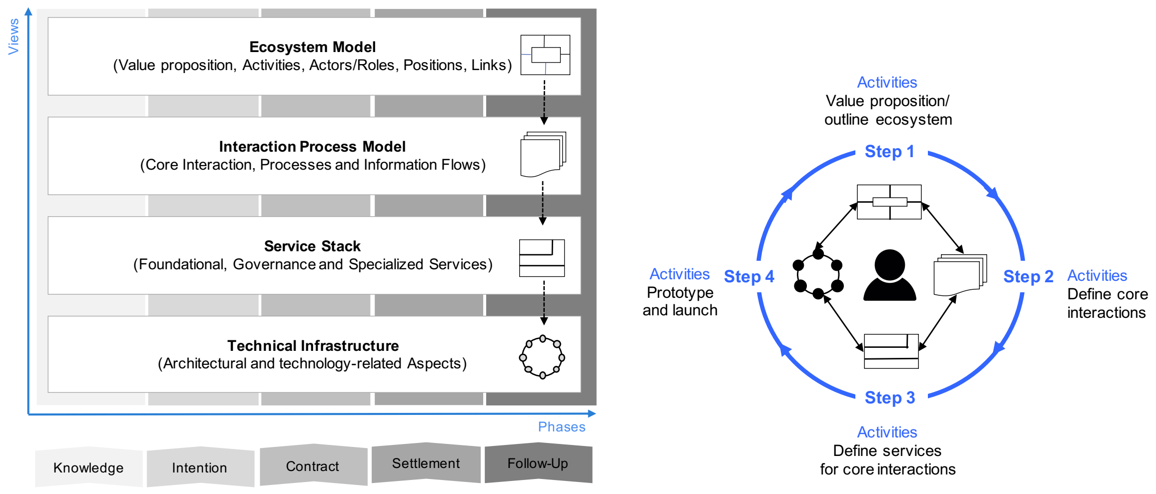
[1] http://www.wemarket.space

- wemarket.space addresses the leisure and cultural landscape of a city. These *city essentials* combine services such as leisure, transportation, and cultural events and are usually offered in any city. As complex services, city essentials differ from city to city in the sense that they can not be standardized since the consumers and their contextual requirements highly personalize them.
- wemarket.space is placed in Frankfurt am Main as an exemplary city. As a case in point, this placement was chosen to demonstrate aspects relevant to the contextual requirements and constraints of city essentials, including, for example, location and other contextual data related to these locations.
- wemarket.spaceis designed and initialized by a core team, which is motivated, and skilled to realize wemarket.space for the city of Frankfurt am Main. Whereas the motivation assumes that they actively contribute to the project, the skills denote the team's ability to fulfill the role.

After describing the application context of the wemarket.space project, the following section shows the application of the reference model for distributed market spaces in order to blueprint and launch an instance of wemarket.space. It first explains the conducted approach, design process, and related activities and continues with the primary outcomes and a detailed description of the wemarket.space elements.

6.1.2 wemarket.space

For the design of the wemarket.space project, the reference model for distributed market spaces was used. The design activities follow the recommendations of the *design phase* of the *life stage model of a reference DMS* (cf. Section 4.9). As shown in Figure 5b, the conducted design process is organized as a wheel model, including the four steps in which an instance of a wemarket.space is blueprinted; that is, when its core entities and aspects are designed. As to Figure 5a, these entities build on each other and are the following: the ecosystem model, the interaction process model, the service stack, and the technical infrastructure.



(a) Reference model for distributed market spaces (b) Design steps and related activities

Figure 6.2: wemarket.space design process

Design steps and conducted activities are:

- Step 1 – to define the value proposition, and based on that, outline the wemarket.space ecosystem structure that embraces actors, roles, and core activity flows they need to align with
- Step 2 – to define wemarket.space core interactions, that is, core processes necessary to enable the interactions among actors within the ecosystem
- Step 3 – to define wemarket.space services required to support the identified interactions on the operational level
- Step 4 – to prototype and launch wemarket.space technical infrastructure required to realize the identified services on the software and hardware level.

The outcomes of each of the conducted steps are summarized in Figure 6.3 and explained in the following.

Step 1 Outlining Ecosystem

The value proposition of the wemarket.space ecosystem is determined by the project goal and formulated as a service ecosystem to enhance the city's intelligence and facilitate seamless consumer experiences. In turn consumer experiences refer to city essentials, which are considered a kind of complex products as defined in Section 6.1.1.

Activities necessary to be conducted by the actors (i.e., participants of the ecosystem) are activities related to the ecosystem foundation and market transaction of city essentials. Actors who are expected to participate and accomplish these activities are inhabitants, including visitors and tourists, business, and other city's stakeholders as institutions, municipalities, and the city's administration. Roles these actors might

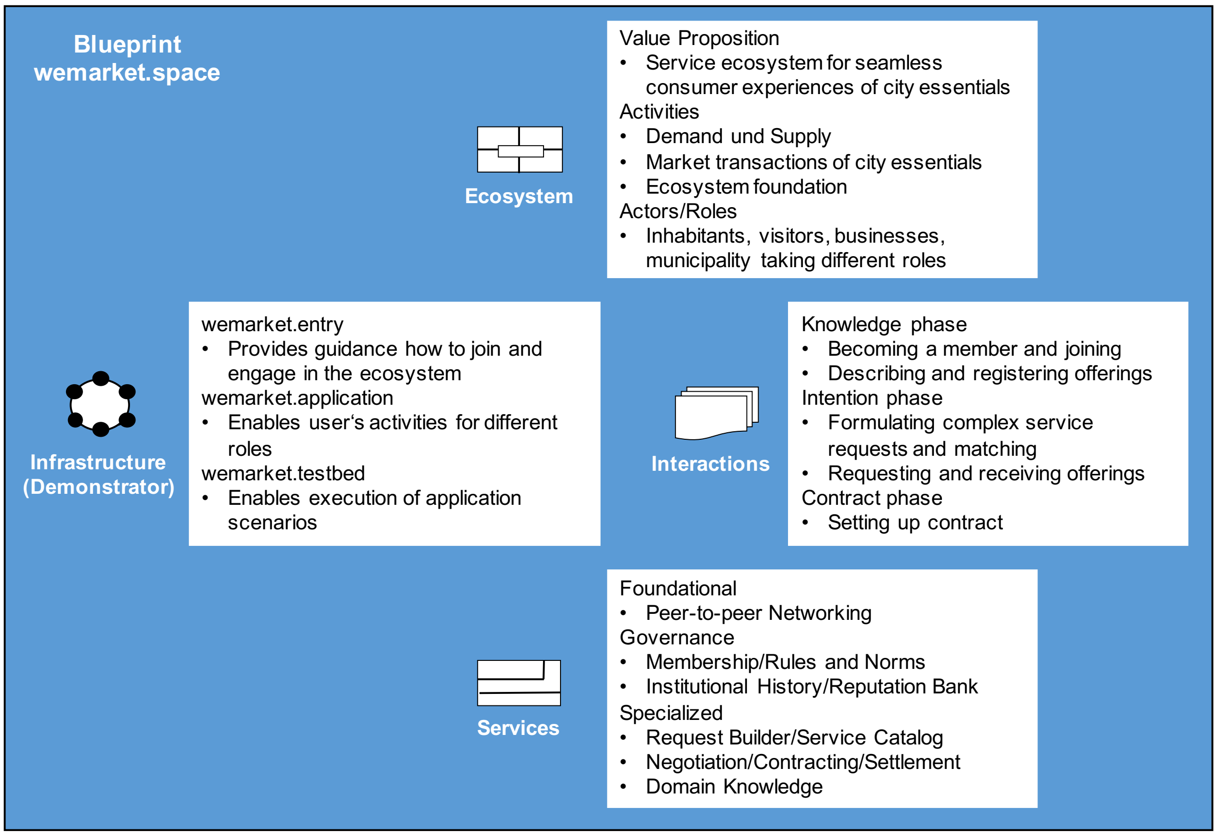

Blueprint wemarket.space
Ecosystem
Value Proposition
• Service ecosystem for seamless consumer experiences of city essentials
Activities
• Demand und Supply
• Market transactions of city essentials
• Ecosystem foundation
Actors/Roles
• Inhabitants, visitors, businesses, municipality taking different roles
Infrastructure (Demonstrator)
wemarket.entry
• Provides guidance how to join and engage in the ecosystem
wemarket.application
• Enables user's activities for different roles
wemarket.testbed
• Enables execution of application scenarios
Interactions
Knowledge phase
• Becoming a member and joining
• Describing and registering offerings
Intention phase
• Formulating complex service requests and matching
• Requesting and receiving offerings
Contract phase
• Setting up contract
Services
Foundational
• Peer-to-peer Networking
Governance
• Membership/Rules and Norms
• Institutional History/Reputation Bank
Specialized
• Request Builder/Service Catalog
• Negotiation/Contracting/Settlement
• Domain Knowledge

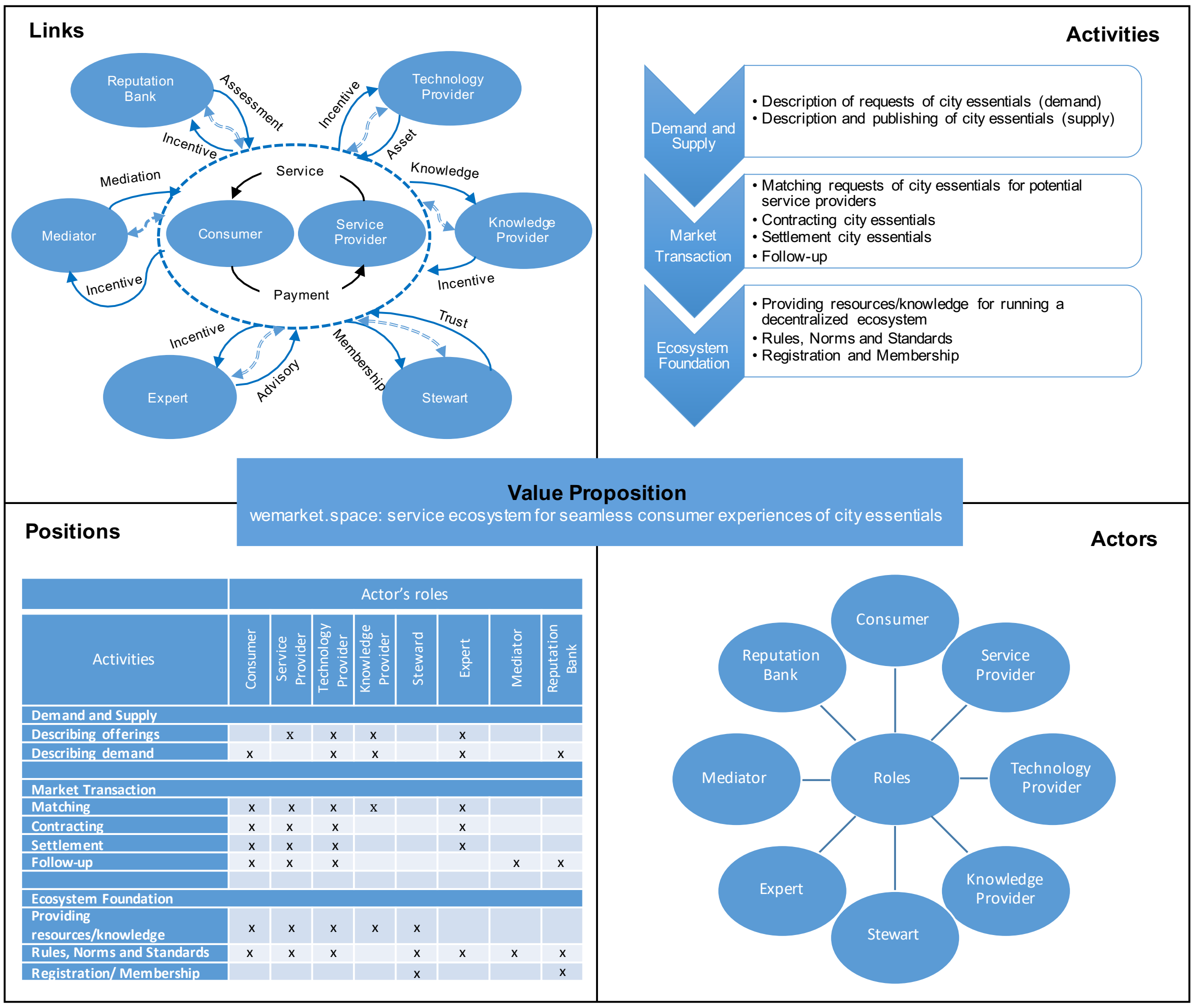
Figure 6.3: Blueprint of wemarket.space summarizing the outcomes of the design process (cf. Figure 6.2)

take are consumer and provider as well as the shaper roles. Realistically, the enabler roles (steward, knowledge provider, and technology provider) will be taken by the core team for the duration of the design stage. This is to ensure a coordinated instantiation of the service environment in the first place, and it is subject to change within the ignition stage. The positions and links are modeled according to considerations above (cf. Section 4.5.3, see Figure 4.5).

Step 2 Defining Core Interactions

For the blueprinting of wemarket.space, three core interactions are chosen to start with. These are the interactions addressing the first three phases of the Interaction Process Model (cf. Section 4.3) and, thus, the knowledge, intention, and contracting phases as highlighted in Figure 6.5. These include joining and becoming a member of the ecosystem and describing and registering offerings (for BPMN diagram, see cf. Figure 4.8).

Furthermore, additional processes include formulating complex service requests, matching potential providers and receiving offerings, and setting up a contract (for BPMN diagrams, see cf. Section 4.6.1., Section 4.6.2, and Section 4.6.3). The settlement of


Links
Reputation Bank
Technology Provider
Assessment
Incentive
Incentive
Asset
Mediation
Service
Knowledge
Mediator
Consumer
Service Provider
Knowledge Provider
Incentive
Payment
Incentive
Trust
Incentive
Membership
Advisory
Expert
Steward
Activities
Demand and Supply
• Description of requests of city essentials (demand)
• Description and publishing of city essentials (supply)
Market Transaction
• Matching requests of city essentials for potential service providers
• Contracting city essentials
• Settlement city essentials
• Follow-up
Ecosystem Foundation
• Providing resources/knowledge for running a decentralized ecosystem
• Rules, Norms and Standards
• Registration and Membership
Value Proposition
wemarket.space: service ecosystem for seamless consumer experiences of city essentials
Positions
Actors
Consumer
Reputation Bank
Service Provider
Mediator
Roles
Technology Provider
Expert
Knowledge Provider
Steward

	Actor's roles							
Activities	Consumer	Service Provider	Technology Provider	Knowledge Provider	Steward	Expert	Mediator	Reputation Bank
Demand and Supply								
Describing offerings		x	x	x		x		
Describing demand	x		x	x		x		x
Market Transaction								
Matching	x	x	x	x		x		
Contracting	x	x	x			x		
Settlement	x	x	x			x		
Follow-up	x	x	x				x	x
Ecosystem Foundation								
Providing resources/knowledge	x	x	x	x	x			
Rules, Norms and Standards	x	x	x		x	x	x	x
Registration/ Membership					x			x

Figure 6.4: Ecosystem model of the wemarket.space ecosystem

contracts is assumed to be a trustful settlement. It relies on direct communication with involved providers and the agreed payment and delivery modalities (e.g., cash, credit card, or online payment modalities). The full-flagged settlement (supporting trustless and trusted third-party settlement), including after-sales activities, will be undertaken in the ignition stage.

Step 3 Defining Services

Based on the chosen core interactions (see Figure 6.5), a set of services is included as shown in Figure 6.6. These are foundational services are necessary for peer-to-peer networking followed by a basic set of governance services for the facility of essential self-governance mechanisms of the ecosystem. In addition, for the implementation of knowledge, intention, and contracting processes, further services are selected. Namely, request builder, service catalog, negotiation, contracting, and domain knowledge (for a detailed description of cf. Section 4.5.).

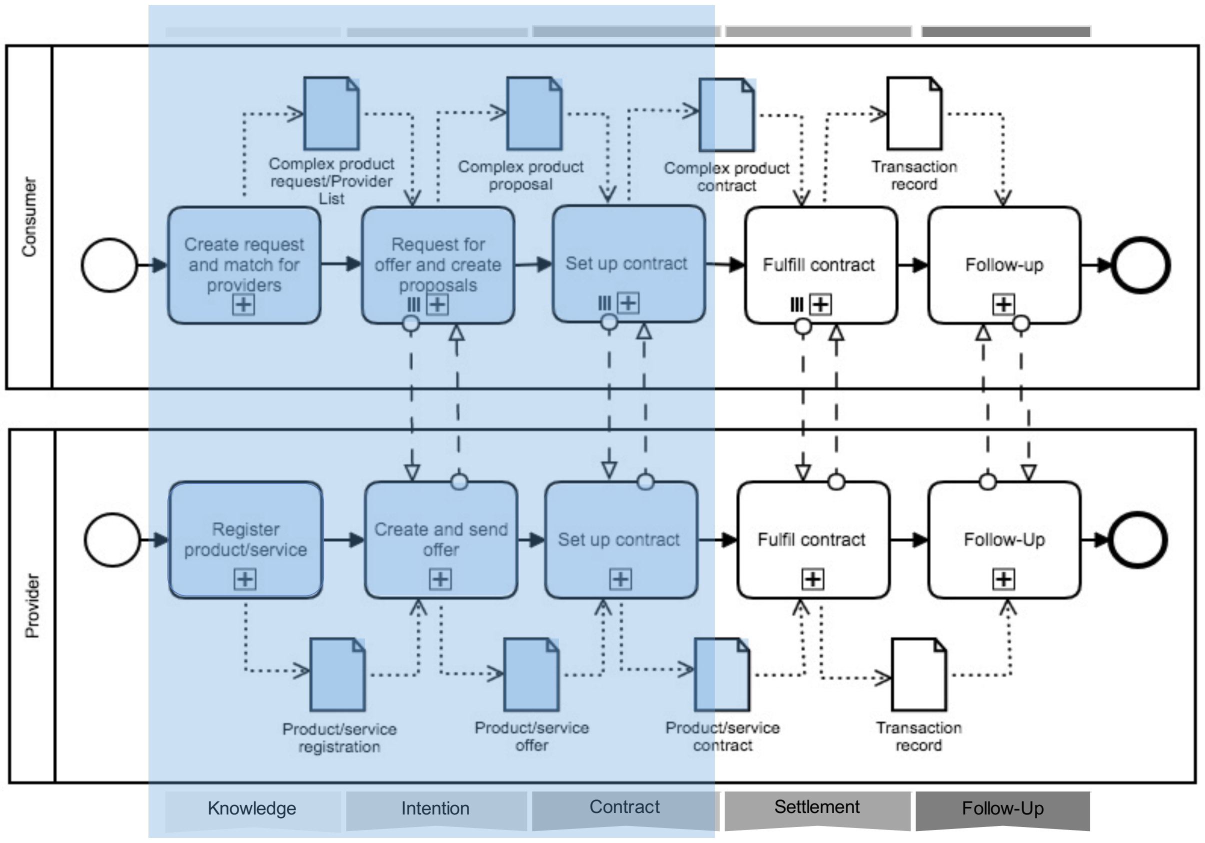


Figure 6.5: Overview of interactions supported by the wemarket.space

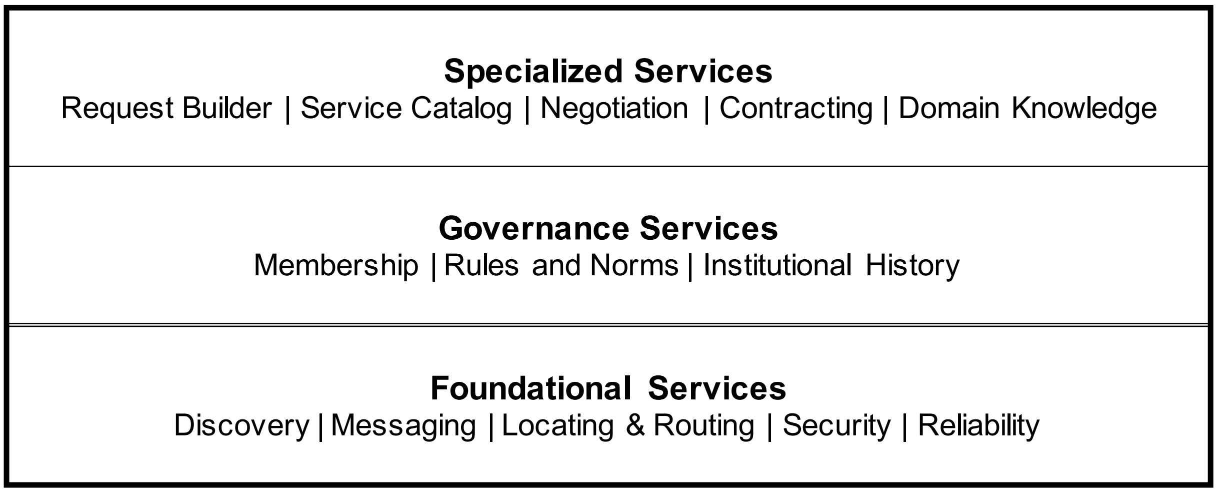

Specialized Services
Request Builder | Service Catalog | Negotiation | Contracting | Domain Knowledge
Governance Services
Membership | Rules and Norms | Institutional History
Foundational Services
Discovery | Messaging | Locating & Routing | Security | Reliability

Figure 6.6: Service stack of the wemarket.space

Step 4 Prototyping and Launching

The infrastructure of the wemarket.space has been prototyped based on the implementation presented in Chapter 5 and launched as a demonstrator in a testbed environment.

The demonstrator is implemented as a web application (JavaScript, HTML, and CSS) as depicted in Figure 6.7. It composes of two parts **wemarket entry** and **wemarket application**. Together these two realize the foundational, governance, and specialized services as specified above and summarized in Figure 6.6.

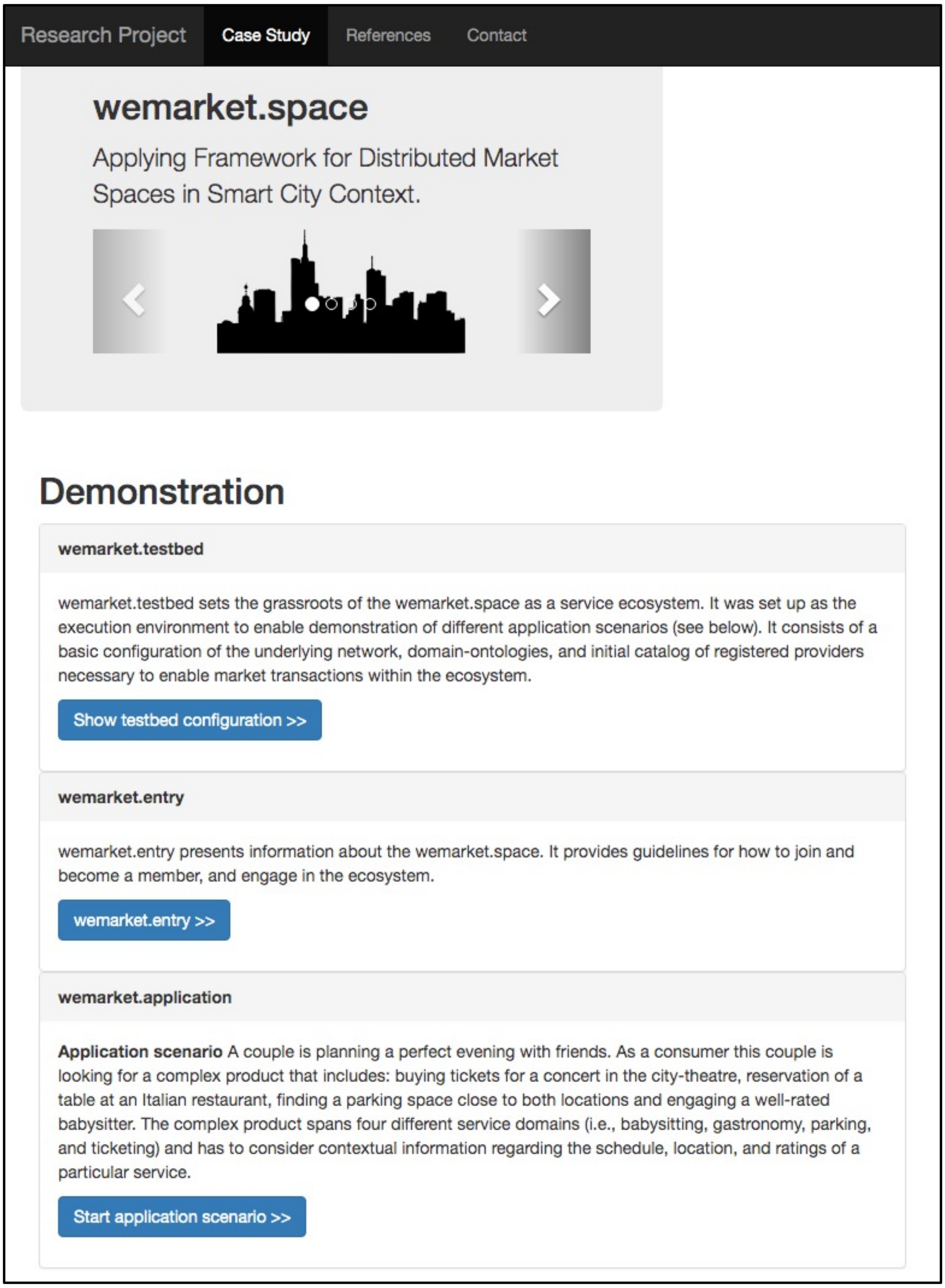
wemarket entry is published under the URL (www.wemarket.space) and denotes a permanently accessible public node. It provides information about wemarket.space and guidelines for joining and participating. It realizes the process of membership and joining the wemarket.space, and thus demonstrates the bootstrapping operation and the formation of the underlying (DHT-based) peer-to-peer network.

wemarket application runs on each participating node represented by a web browser instance. It enables user activity for consumer and provider roles currently supporting only human interfaces. It supports the handling of city essentials scenarios and thus demonstrates the market exchange via wemarket.space as a service ecosystem.

wemarket testbed serves two purposes. First, it yields the foundation for bootstrapping wemarket.space as a peer-to-peer based ecosystem. Secondly, it yields the basic knowledge required for the market exchange among ecosystem participants. It includes:

- **a basic network configuration** – which was set up to ensure a sufficient number of members is available so that tasks of the bootstrapping operation can be accomplished and new users enabled to join. The basic network is configured as a structured overlay containing 16 peers (ring topology implementing Chord [Sto+01], cf. Section 5.6.1).
- **a basic set of domain ontologies** for describing city essentials. These include ontologies for domain ticketing, gastronomy, parking, and babysitting, as these domains are relevant to the demonstration scenario, which is discussed in more detail below. In addition, these ontologies are used to describe user interfaces generated by the mimesis service as the implementation of the CPR Builder component (cf., Section 5.6.1).
- **an initial catalog of providers** – which was instantiated to enable the initial matching for providers. Catalog of providers have been generated following the concepts defined by the DMS Ontology (cf. 5.5.1). The generated data draw on existing entries (e.g., yellow pages, commercial register for Frankfurt City) and represent a mix of private and public companies for afore-mentioned domains. Each of them is considered a member of wemarket.space with a unique MemberID and is registered for the provider role.

For a detailed description of the wemarket testbed configuration see Listings in Appendix A.1 and Appendix A.2.



wemarket.space

Applying Framework for Distributed Market Spaces in Smart City Context.

Demonstration

wemarket.testbed

wemarket.testbed sets the grassroots of the wemarket.space as a service ecosystem. It was set up as the execution environment to enable demonstration of different application scenarios (see below). It consists of a basic configuration of the underlying network, domain-ontologies, and initial catalog of registered providers necessary to enable market transactions within the ecosystem.

wemarket.entry

wemarket.entry presents information about the wemarket.space. It provides guidelines for how to join and become a member, and engage in the ecosystem.

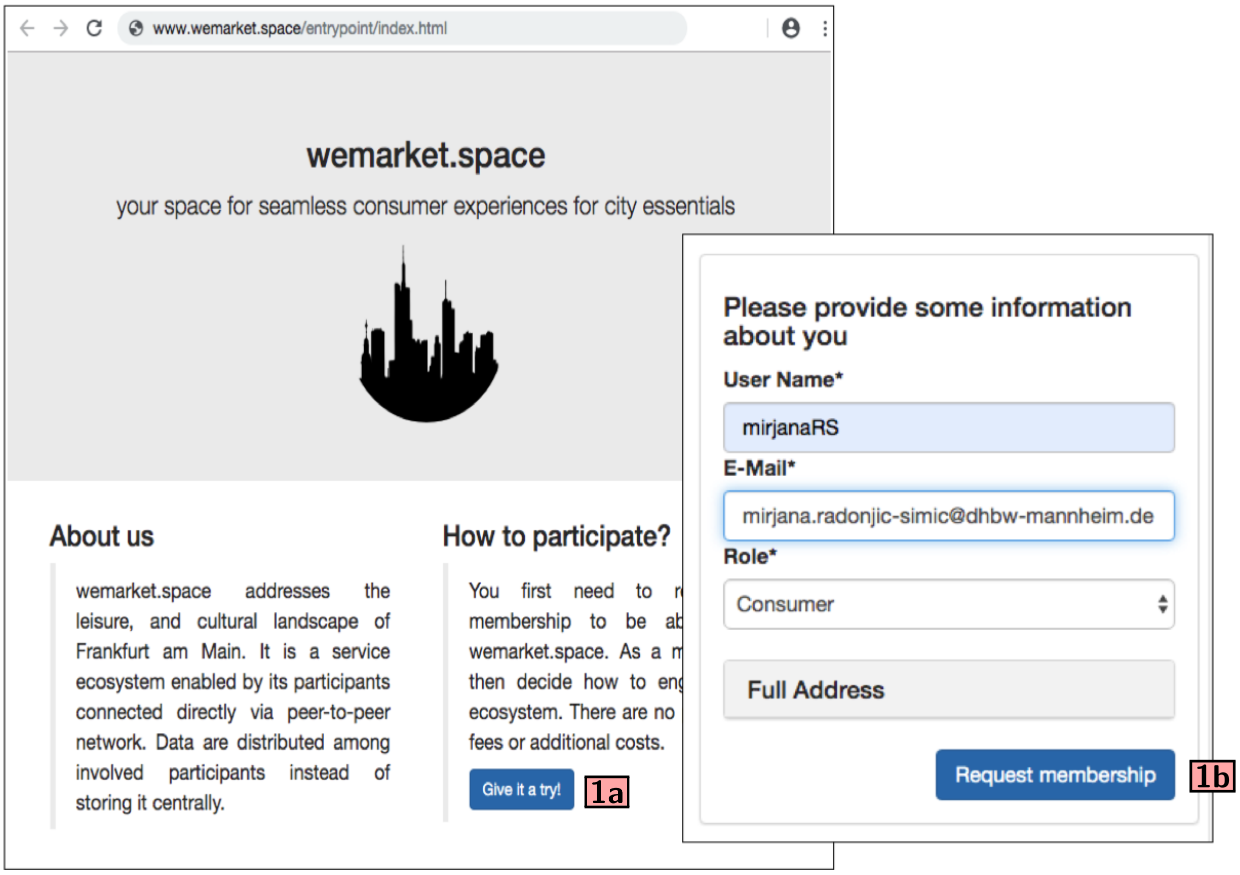
wemarket.application

Application scenario A couple is planning a perfect evening with friends. As a consumer this couple is looking for a complex product that includes: buying tickets for a concert in the city-theatre, reservation of a table at an Italian restaurant, finding a parking space close to both locations and engaging a well-rated babysitter. The complex product spans four different service domains (i.e., babysitting, gastronomy, parking, and ticketing) and has to consider contextual information regarding the schedule, location, and ratings of a particular service.

Figure 6.7: wemarket.space demonstrator implemented as a web application.

6.1.3 Demonstration

After an instance of wemarket has been designed, prototyped, and launched, this section demonstrates its core interactions taking the consumer's perspective. For an exemplary city essential scenario, recall the use case of our couple planning an evening with friends. As explained in Section 1.1, our couple is looking for a personalized bundle of services. It includes tickets for a concert, reservation of a table at an Italian restaurant, finding a parking spot close to both locations, and engaging a well-rated babysitter to watch after children.



www.wemarket.space/entrypoint/index.html

wemarket.space

your space for seamless consumer experiences for city essentials

About us

wemarket.space addresses the leisure, and cultural landscape of Frankfurt am Main. It is a service ecosystem enabled by its participants connected directly via peer-to-peer network. Data are distributed among involved participants instead of storing it centrally.

How to participate?

You first need to r
membership to be ab
wemarket.space. As a m
then decide how to eng
ecosystem. There are no
fees or additional costs.

Give it a try! 1a

Please provide some information about you

User Name*: mirjanaRS

E-Mail*: mirjana.radonjic-simic@dhbw-mannheim.de

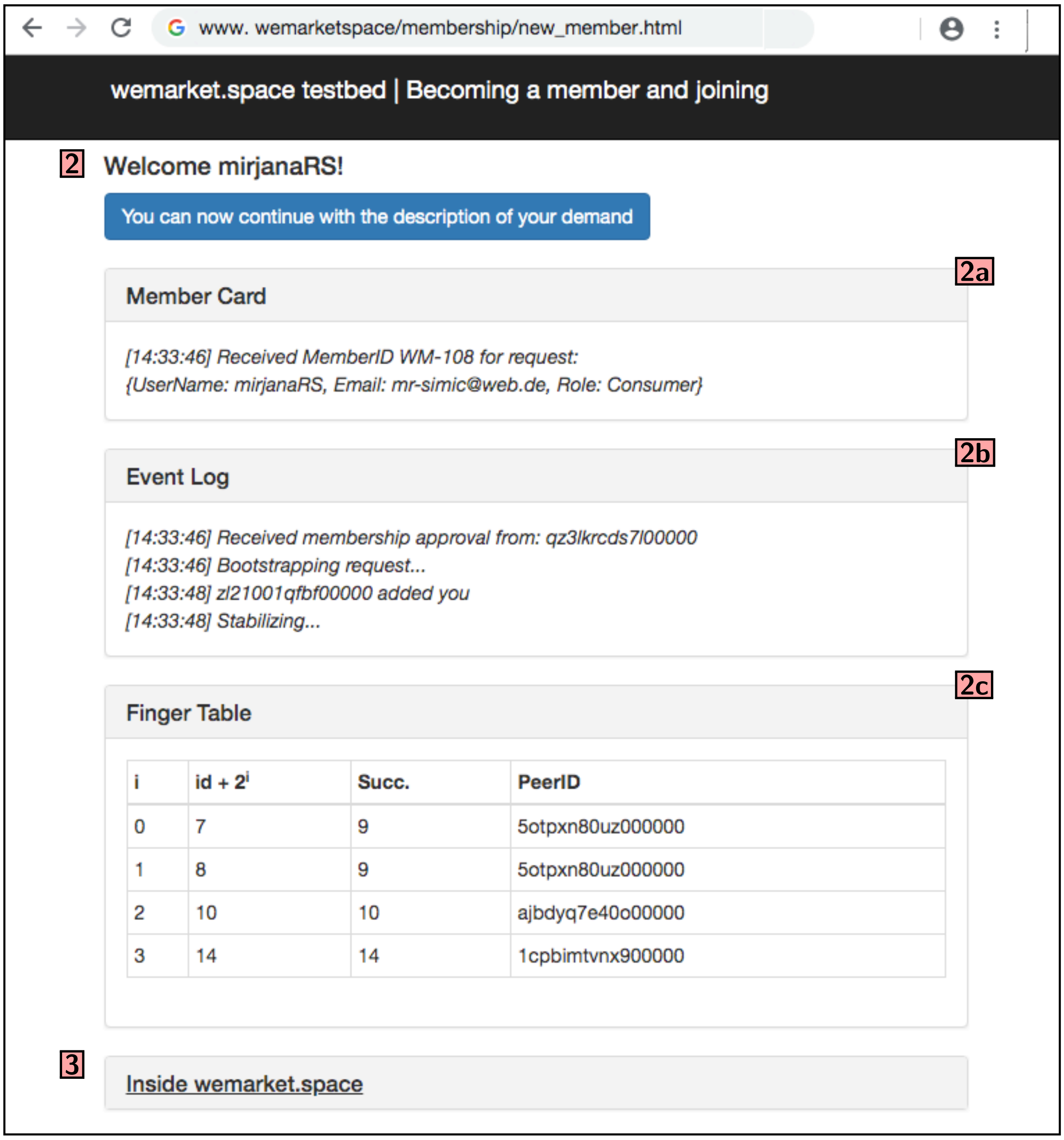
Role*: Consumer

Full Address

Request membership 1b

Figure 6.8: wemarket.space entry point

Their demand spans four different service domains (i.e., ticketing, gastronomy, parking, and babysitting) and has to consider contextual information regarding the schedule, location, and ratings of a particular service.


www. wemarketspace/membership/new_member.html
2
2a
2b
2c
3
You can now continue with the description of your demand
Inside wemarket.space

wemarket.space testbed | Becoming a member and joining

Welcome mirjanaRS!

Member Card

[14:33:46] Received MemberID WM-108 for request:
{UserName: mirjanaRS, Email: mr-simic@web.de, Role: Consumer}

Event Log

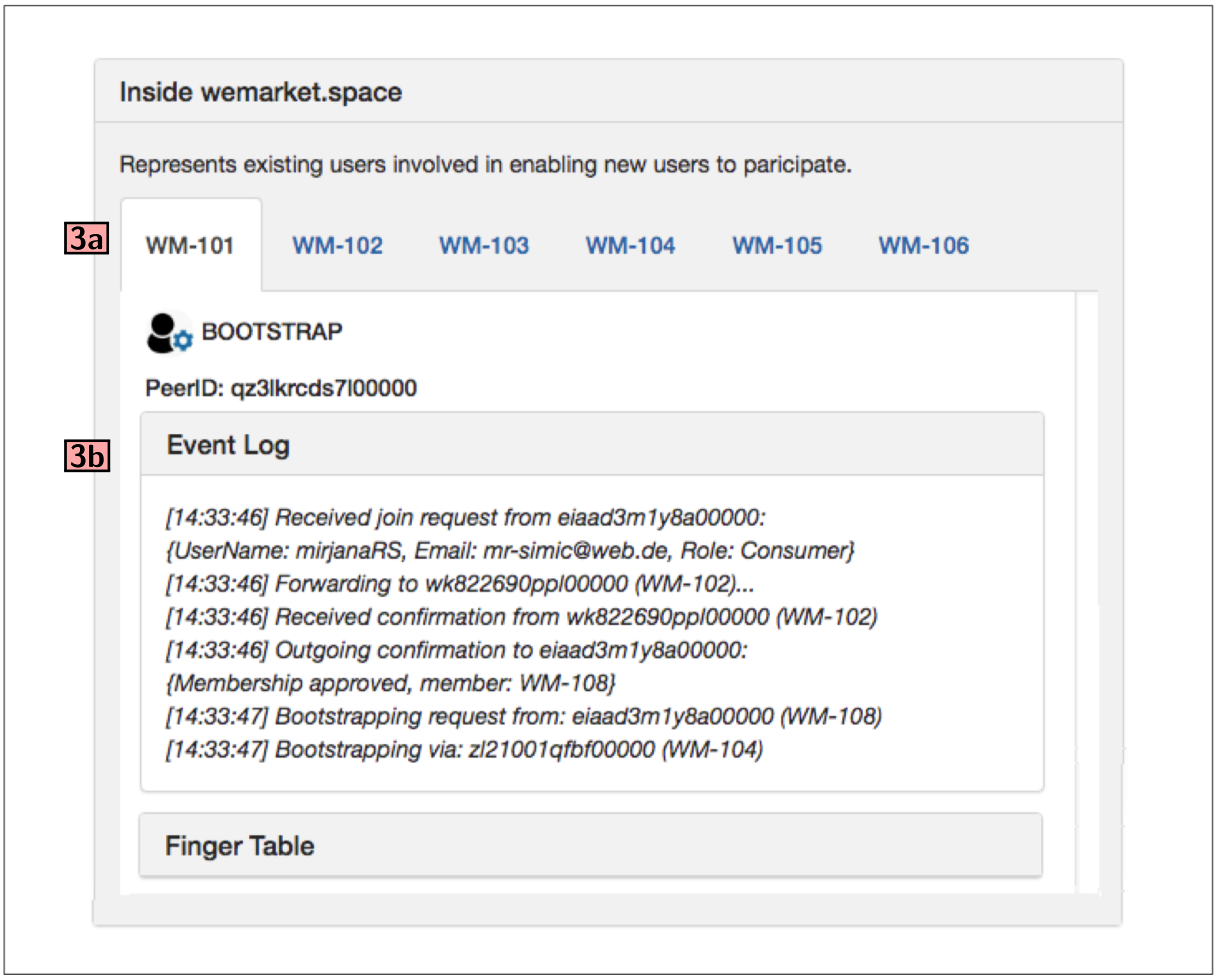
```
[14:33:46] Received membership approval from: qz3lkrcds7l00000
[14:33:46] Bootstrapping request...
[14:33:48] zl21001qfbf00000 added you
[14:33:48] Stabilizing...
```

Finger Table

i	id + 2^i	Succ.	PeerID
0	7	9	5otpxn80uz000000
1	8	9	5otpxn80uz000000
2	10	10	ajbdyq7e40o00000
3	14	14	1cpbimtvnx900000

Figure 6.9: wemarket.space – new-user bootstrapped

In order to start planing, our couple first needs to become a member by visiting the wemarket entry point (see 1a in Figure 6.8). As a new user, this couple needs to provide a few pieces of information and submit a request for membership 1b. This request is sent to the network, that is, to the ecosystem's users responsible for the processing of membership requests (i.e., steward role) and ultimately bootstrapping (see Figure 6.10 for interactions inside wemarket.space).

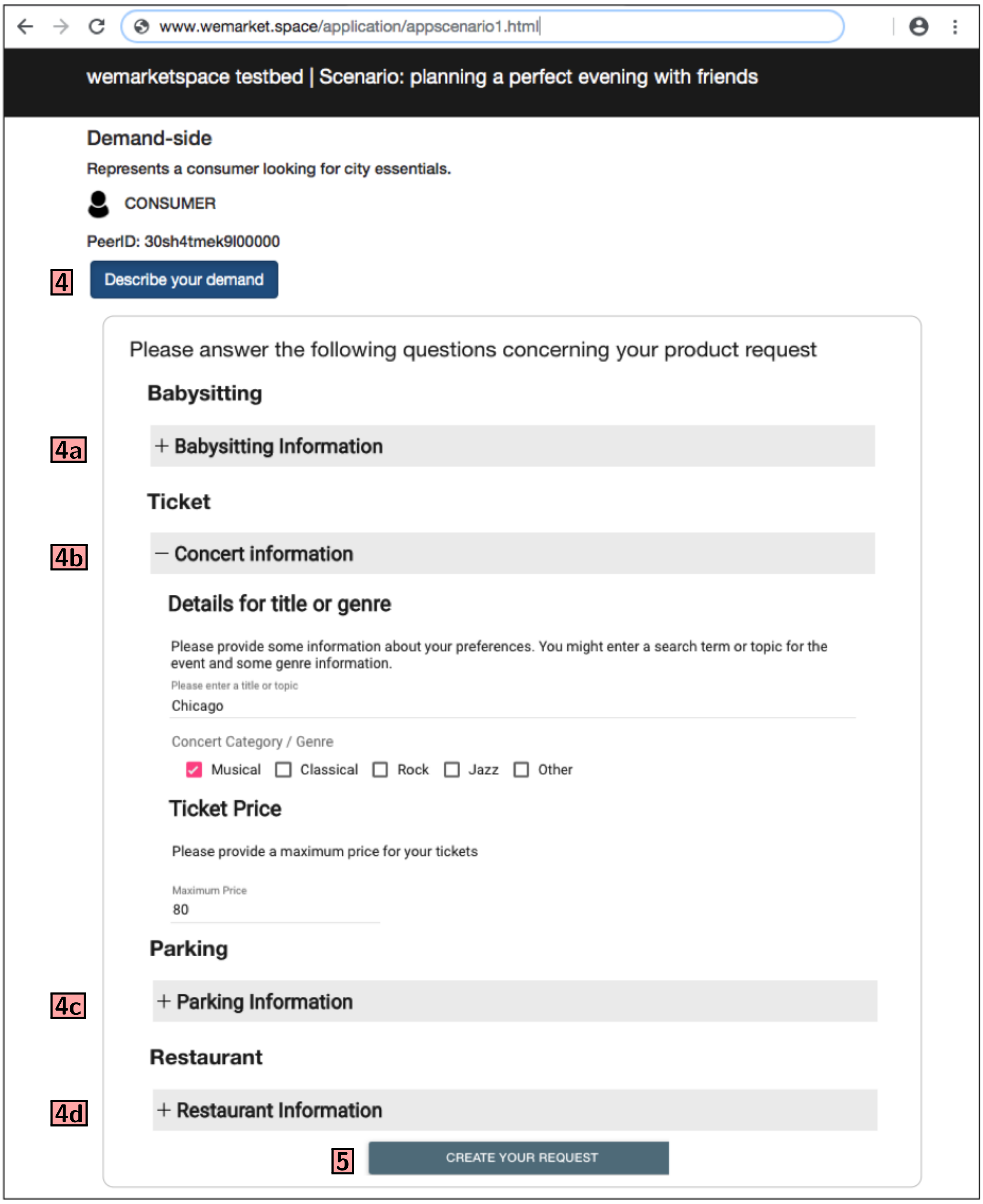


```
[14:33:46] Received join request from eiaad3m1y8a00000:
{UserName: mirjanaRS, Email: mr-simic@web.de, Role: Consumer}
[14:33:46] Forwarding to wk822690ppl00000 (WM-102)...
[14:33:46] Received confirmation from wk822690ppl00000 (WM-102)
[14:33:46] Outgoing confirmation to eiaad3m1y8a00000:
{Membership approved, member: WM-108}
[14:33:47] Bootstrapping request from: eiaad3m1y8a00000 (WM-108)
[14:33:47] Bootstrapping via: zl21001qfbf00000 (WM-104)
```

Figure 6.10: wemarket.space – new-user bootstrapped (inside wemarket.space)

As a result, our couple becomes a member 2 by getting a member card as the identification within the ecosystem 2a, and a MemberID 2b necessary for the direct interactions among the network 2c. This is depicted in Figure 6.9. Area 3 provides a view behind the scene. That is an inside of wemarket.space showing existing users within the network who are involved in enabling new users to join and participate 3a. Figure 6.10 illustrates how they propagated membership requests and thus accomplished the bootstrapping procedure 3b.

As a new user, our couple can continue with the description of the concrete demand 4 as shown in Figure 6.11. In this case, the selection of domains describes the desired service combination 4a, 4b, 4c, 4d. Thereby for each of the services, different fields have to be entered. For example as in 4b a ticket for Chicago Musical, described by name, category/genre, and price.


www.wemarket.space/application/appscenario1.html

wemarketspace testbed | Scenario: planning a perfect evening with friends

Demand-side

Represents a consumer looking for city essentials.

CONSUMER

PeerID: 30sh4tmek9l00000

4 Describe your demand

Please answer the following questions concerning your product request

Babysitting

4a + Babysitting Information

Ticket

4b − Concert information

Details for title or genre

Please provide some information about your preferences. You might enter a search term or topic for the event and some genre information.

Please enter a title or topic: Chicago

Concert Category / Genre: [✓ Musical] Classical Rock Jazz Other

Ticket Price

Please provide a maximum price for your tickets

Maximum Price: 80

Parking

4c + Parking Information

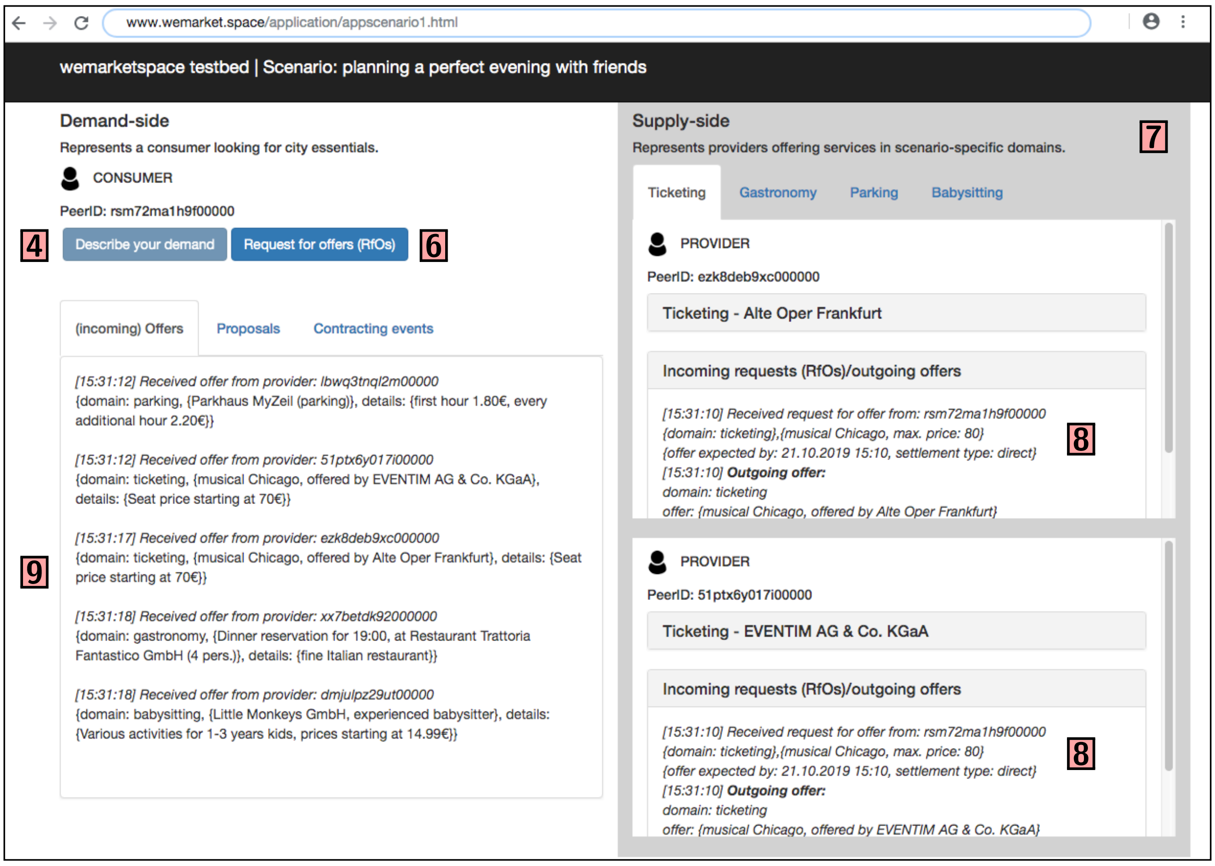
Restaurant

4d + Restaurant Information

5 CREATE YOUR REQUEST

Figure 6.11: wemarket.space – description of demand, data entering

By submitting 5, the complex product request is created (the resulting RDF document can be seen in Appendix A.3). The created request is then published within the network by clicking the button Request for Offers (RfO) 6.



wemarketspace testbed | Scenario: planning a perfect evening with friends

Demand-side

Represents a consumer looking for city essentials.

CONSUMER

PeerID: rsm72ma1h9f00000

[15:31:12] Received offer from provider: lbwq3tnql2m00000
{domain: parking, {Parkhaus MyZeil (parking)}, details: {first hour 1.80€, every additional hour 2.20€}}

[15:31:12] Received offer from provider: 51ptx6y017i00000
{domain: ticketing, {musical Chicago, offered by EVENTIM AG & Co. KGaA}, details: {Seat price starting at 70€}}

[15:31:17] Received offer from provider: ezk8deb9xc000000
{domain: ticketing, {musical Chicago, offered by Alte Oper Frankfurt}, details: {Seat price starting at 70€}}

[15:31:18] Received offer from provider: xx7betdk92000000
{domain: gastronomy, {Dinner reservation for 19:00, at Restaurant Trattoria Fantastico GmbH (4 pers.)}, details: {fine Italian restaurant}}

[15:31:18] Received offer from provider: dmjulpz29ut00000
{domain: babysitting, {Little Monkeys GmbH, experienced babysitter}, details: {Various activities for 1-3 years kids, prices starting at 14.99€}}

Supply-side

Represents providers offering services in scenario-specific domains.

PROVIDER

PeerID: ezk8deb9xc000000

Ticketing - Alte Oper Frankfurt

Incoming requests (RfOs)/outgoing offers

[15:31:10] Received request for offer from: rsm72ma1h9f00000
{domain: ticketing},{musical Chicago, max. price: 80}
{offer expected by: 21.10.2019 15:10, settlement type: direct}
*[15:31:10] **Outgoing offer:***
domain: ticketing
offer: {musical Chicago, offered by Alte Oper Frankfurt}

PROVIDER

PeerID: 51ptx6y017i00000

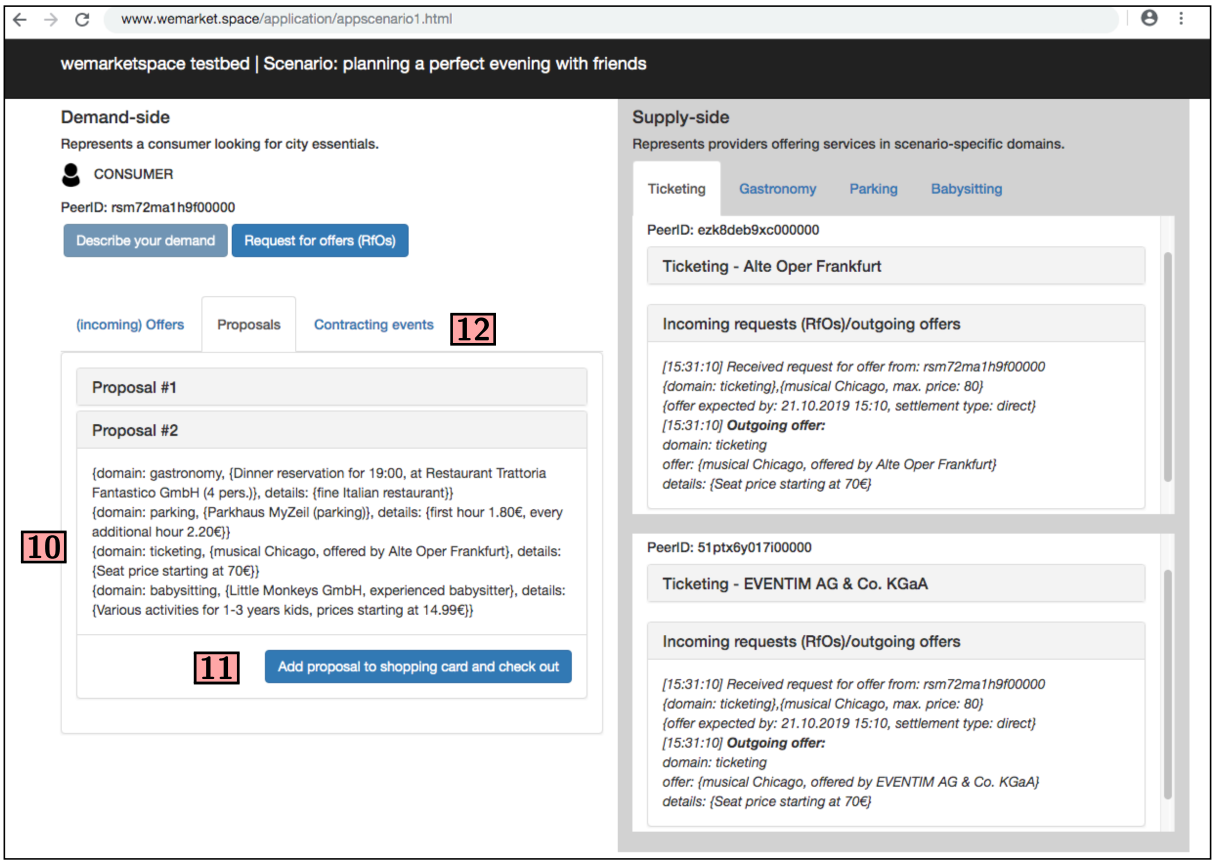
Ticketing - EVENTIM AG & Co. KGaA

Incoming requests (RfOs)/outgoing offers

[15:31:10] Received request for offer from: rsm72ma1h9f00000
{domain: ticketing},{musical Chicago, max. price: 80}
{offer expected by: 21.10.2019 15:10, settlement type: direct}
*[15:31:10] **Outgoing offer:***
domain: ticketing
offer: {musical Chicago, offered by EVENTIM AG & Co. KGaA}

Figure 6.12: wemarket.space – matching, requesting for offers, and receiving offers

This starts the matching and, thus, interactions of the consumer with potential providers for demanded services. This is illustrated in Figure 6.12. The right side of Figure 6.12 represents the addressed providers 7, which are grouped by the domains in which they offer services of interest. Also, incoming requests for offers and outgoing offers to the requesting consumer are shown 8. On the consumer side, all received offers are presented in the tab (incoming) Offers 9, and based on that, viable proposals are created and ordered in the adjacent tab (see 10 in Figure 6.13). By accepting a particular proposal 11, the process of setting up an umbrella contract is started, and with that the realization of the two-stage contract confirmation process. After all pending contracts have been confirmed, the consumer can finish the process of creating a legally binding contract, and each involved provider then receives the order confirmation as the legally binding agreement.


www.wemarket.space/application/appscenario1.html

wemarketspace testbed | Scenario: planning a perfect evening with friends

Demand-side

Represents a consumer looking for city essentials.

CONSUMER

PeerID: rsm72ma1h9f00000

Describe your demand | Request for offers (RfOs)

(incoming) Offers | Proposals | Contracting events 12

Proposal #1

Proposal #2

10 {domain: gastronomy, {Dinner reservation for 19:00, at Restaurant Trattoria Fantastico GmbH (4 pers.)}, details: {fine Italian restaurant}}
{domain: parking, {Parkhaus MyZeil (parking)}, details: {first hour 1.80€, every additional hour 2.20€}}
{domain: ticketing, {musical Chicago, offered by Alte Oper Frankfurt}, details: {Seat price starting at 70€}}
{domain: babysitting, {Little Monkeys GmbH, experienced babysitter}, details: {Various activities for 1-3 years kids, prices starting at 14.99€}}

11 Add proposal to shopping card and check out

Supply-side

Represents providers offering services in scenario-specific domains.

Ticketing | Gastronomy | Parking | Babysitting

PeerID: ezk8deb9xc000000

Ticketing - Alte Oper Frankfurt

Incoming requests (RfOs)/outgoing offers

[15:31:10] Received request for offer from: rsm72ma1h9f00000
{domain: ticketing},{musical Chicago, max. price: 80}
{offer expected by: 21.10.2019 15:10, settlement type: direct}
[15:31:10] ***Outgoing offer:***
domain: ticketing
offer: {musical Chicago, offered by Alte Oper Frankfurt}
details: {Seat price starting at 70€}

PeerID: 51ptx6y017i00000

Ticketing - EVENTIM AG & Co. KGaA

Incoming requests (RfOs)/outgoing offers

[15:31:10] Received request for offer from: rsm72ma1h9f00000
{domain: ticketing},{musical Chicago, max. price: 80}
{offer expected by: 21.10.2019 15:10, settlement type: direct}
[15:31:10] ***Outgoing offer:***
domain: ticketing
offer: {musical Chicago, offered by EVENTIM AG & Co. KGaA}
details: {Seat price starting at 70€}

Figure 6.13: wemarket.space – creating proposals and starting contracting

This is represented in the tab contracting events 12 on the consumer side and in incomming requests tab on the provider side 8. Providers can confirm the pending contract by clicking on the button "confirm". Such confirmations are to be automated, but for this demonstration, the confirmation is considered a manual activity. All received contract confirmations are listed in 12 presenting the status of the contracting process. After all pending contacts are confirmed, the consumer can finish the process of creating a legally binding contract for each of the offers, and each provider then receives the order confirmation as the legally binding agreement. The legally binding agreement will then be executed by the payment on the consumer side, providing access to the services bought on the provider side. In this case, for example, a QR code for the tickets and a confirmation for the babysitter.

6.2 COVID-19 Pandemic Management

The COVID-19 pandemic caused by SARS-CoV-2 has massively impacted the health of many people worldwide since 2019, and still poses significant challenges for social, economic, and political life – in Germany, Europe, and worldwide. Different states use different strategies to tackle the COVID-19 pandemic and return to a "normal" life.

In doing so, they are often pairing measures including temporary lockdowns, massive testing, and vaccines to reach a certain percentage of population immunization [Deb+20].

In Germany, the vaccination strategy is formulated in the "National Vaccination Strategy COVID-19" document [Bun20], and legally stipulated by the Infection Protection Act[Bun]. Accordingly, immunization (i.e., the vaccination process to fight the SARS-COVID-19 virus) must be defined and planed on the national level. However, based on the Infection Protection Act and the inherently federal organization of the German state, the activities of the vaccination process have to be implemented decentrally by the sixteen federal states. This also applies to the surveillance and monitoring of vaccination rates, activities where various parties need to be involved and engaged in coordinating their activities and sharing data that goes beyond the vaccination process. In addition, all activities must comply with the national data protection legislation and the European General Data Protection Regulation (EU-GDPR) [Wol18]. This ensures that the law is applied, as data monitoring and evaluation over the years or even decades goes hand in hand with corresponding long-term data storage.

However, Germany is only a case in point. Governmental decisions always need to be implemented by regional and/or local actors, the number of which varies greatly depending on the country. In addition to the fragmented landscape of actors involved at the national and federal levels, like in Germany, the complexity of the vaccination process imposes additional challenges on the organizational models and underlying information systems needed for implementation. When centrally made governmental decisions need to be implemented by regional and/or local actors, these actors often find themselves in a situation where they need to create either new solutions to implement these decisions, or use existing applications of their choice that they modify for this purpose. The result is a myriad of different solutions. Relevant German solutions either take into account the spread of infections, but do not support activities of the vaccination process, e.g., DEMIS [gem21] and SOMAS [Hel21], In parallel to that, or cover only some aspects, like tracing infected people, e.g., CORONA Warn-App [Rob21], luca-app [cul21], and digital vaccine certificates, e.g., Digital Green Certificate (DGC) [Eur21], or they concentrate on providing physical infrastructure for the exchange of infection-related data, e.g., eDMP [Kas21].

Therefore, there are calls from academia, healthcare professionals, politics, and ultimately the World Health Organization [2] for integrated solutions that bring all the essential parties together and enable them to achieve the common goal of sustainable pandemic control [Wor20]. To this effect, one approach uses digital platforms as technology-based organizational models to cope with the fragmented and complex healthcare scenarios [Neu+20; Alh21]. Some examples are the platforming of healthcare in China that is built upon Tencent's network of healthcare players [Neu+20], and the UniPlat initiative [Seh21] – a program aiming at digitally transforming all healthcare-related services in the Kingdom of Saudi Arabia. However, these are centrally managed and orchestrated platforms that are also less suitable to meet the EU-GDPR and related

[2]https://www.who.int

restrictions, especially when management and storage of personal data are considered.

6.2.1 Application Scenario

The application scenario in COVID-19 Pandemic Management is represented by the goals, scope, and assumptions of the COVID-19-Vacc project. COVID-19-Vacc is an open and decentralized platform model aiming to support the COVID-19 pandemic management in Germany.

Goal of the application scenario: The main goal of the COVID-19-Vacc platform is to address primary aspects of the National Vaccination Strategy COVID-19 and facilitate its implementation as per the Infection Protection Act. Furthermore, such a COVID-19-Vacc platform, which is focused on vaccinations as a starting point, should connect various actors and enables them to involve, conduct, and track the vaccination process while meeting all necessary data protection and security requirements.

Scope of the application scenario: The scope of COVID-19-Vacc platform includes design activities necessary to blueprint a COVID-19-Vacc platform in a way to define how the COVID-19-Vacc works on the organization level, on the level of processes and how it works of on the operational level, thus determining how it facilitates the above organizational and process models.

Assumptions of the application scenario: National Vaccination Strategy COVID-19 defines the main assumptions underlying the COVID-19-Vacc platform. For the implementation of the National Vaccination Strategy COVID-19, three aspects are considered essential [Bun20]. Firstly, the decentralized implementation that is triggered by the federal organization of the German state. According to the Infection Protection Act, its sixteen states are responsible for implementing the vaccination strategy. The second important aspect is implementing the surveillance and monitoring of the vaccination rate– two essential processes planned centrally but must be decentralized implemented. The third aspect is national and European data protection regulations, ensuring that all activities of the vaccination process are implemented accordingly [Lan21].

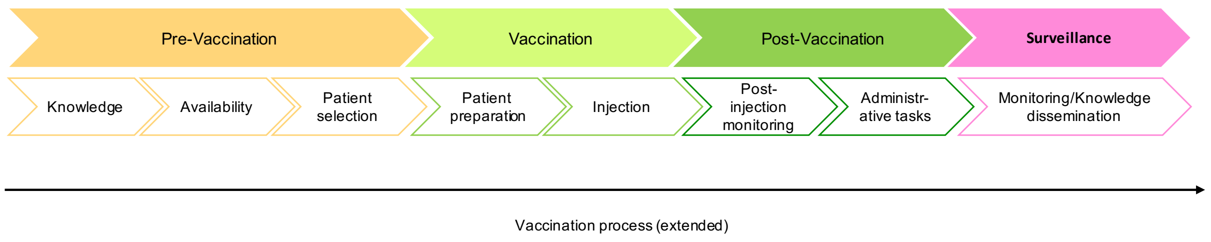


Figure 6.14: Extended vaccination process (based on [Wio+19])

Additional assumptions derive from the requirements regarding the extended vaccination process for the immunization against the SARS-CoV-2 virus. As illustrated in

6.14 it follows the standardized method of the vaccination journey [Wio+19] that includes activities required to administer a vaccine considering necessary medical and administrative standards and norms, but also includes an additional phase for a timely evaluation and active monitoring as required by the COVID-19 vaccination strategy. Accordingly, the extended vaccination process spans four main phases:

- Pre-Vaccination phase – includes activities in which the knowledge about available vaccines is acquired, the availability of a certain number of vaccines ensured, and patients (i.e., individuals to be vaccinated) are selected in terms of scheduling a vaccination appointment.
- Vaccination phase – includes activities in which the patient is prepared for vaccination, informed about potential side effects and adverse events, and the vaccination process, including injection, is executed.
- Post-Vaccination phase – includes activities in which post-injection monitoring is conducted, and post-procedure administrative tasks are completed. This involves recording vaccination-related data and any side effects that have been occurred during the vaccination procedure, issuing vaccination certificates and other administrative tasks like billing or data transmission.
- Surveillance phase – includes activities in which an active data-based monitoring and surveillance of the vaccination rate is possible. These are activities related to monitoring and tasks relevant for permanent risk assessments and knowledge dissemination regarding clinical studies and statistics on adverse events and side effects.

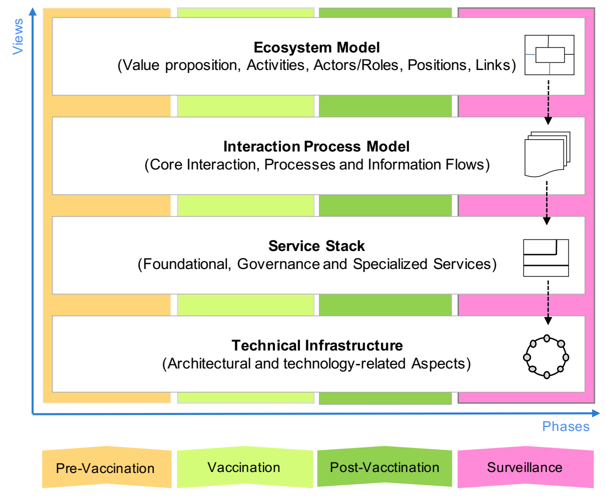
6.2.2 COVID-19-Vacc

After setting the goal, scope, and assumptions of the COVID-19-Vacc platform, this section presents how the COVID-19-Vacc platform was blueprinted using the adapted and adjusted reference model for distributed spaces (see Figure 6.15a).

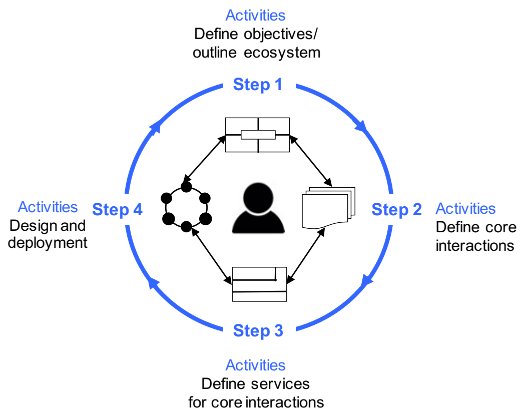
The blueprinting activities follow the recommendations of the *design phase* of the *life stage model of a reference DMS* (cf. Section 4.9). As shown in Figure 6.15a, the design phase includes modeling activities in four views considering the adjusted phase model. The adjustment addresses the specificities of the extended vaccination process (cf. Figure 6.14), and thus spans the phases pre-vaccination, vaccination, post-vaccination, and surveillance. Due to the application scope defined above, the blueprinting of a COVID-19-Vacc concentrate on the activities of the first three steps:

- Step 1 – to define the main objectives of a COVID-19-Vacc, and based on that, to outline the COVID-19-Vacc:Ecosystem structure that embraces actors, roles, and core activity flows they need to align with
- Step 2 – to define COVID-19-Vacc:Interactions, core processes necessary to enable the interactions among actors within the ecosystem

- Step 3 – to define COVID-19-Vacc:Services required to support the identified interactions on the operational level


Views
Ecosystem Model
(Value proposition, Activities, Actors/Roles, Positions, Links)
Interaction Process Model
(Core Interaction, Processes and Information Flows)
Service Stack
(Foundational, Governance and Specialized Services)
Technical Infrastructure
(Architectural and technology-related Aspects)
Phases
Pre-Vaccination
Vaccination
Post-Vacctination
Surveillance

(a) Adapted reference model for distributed market spaces

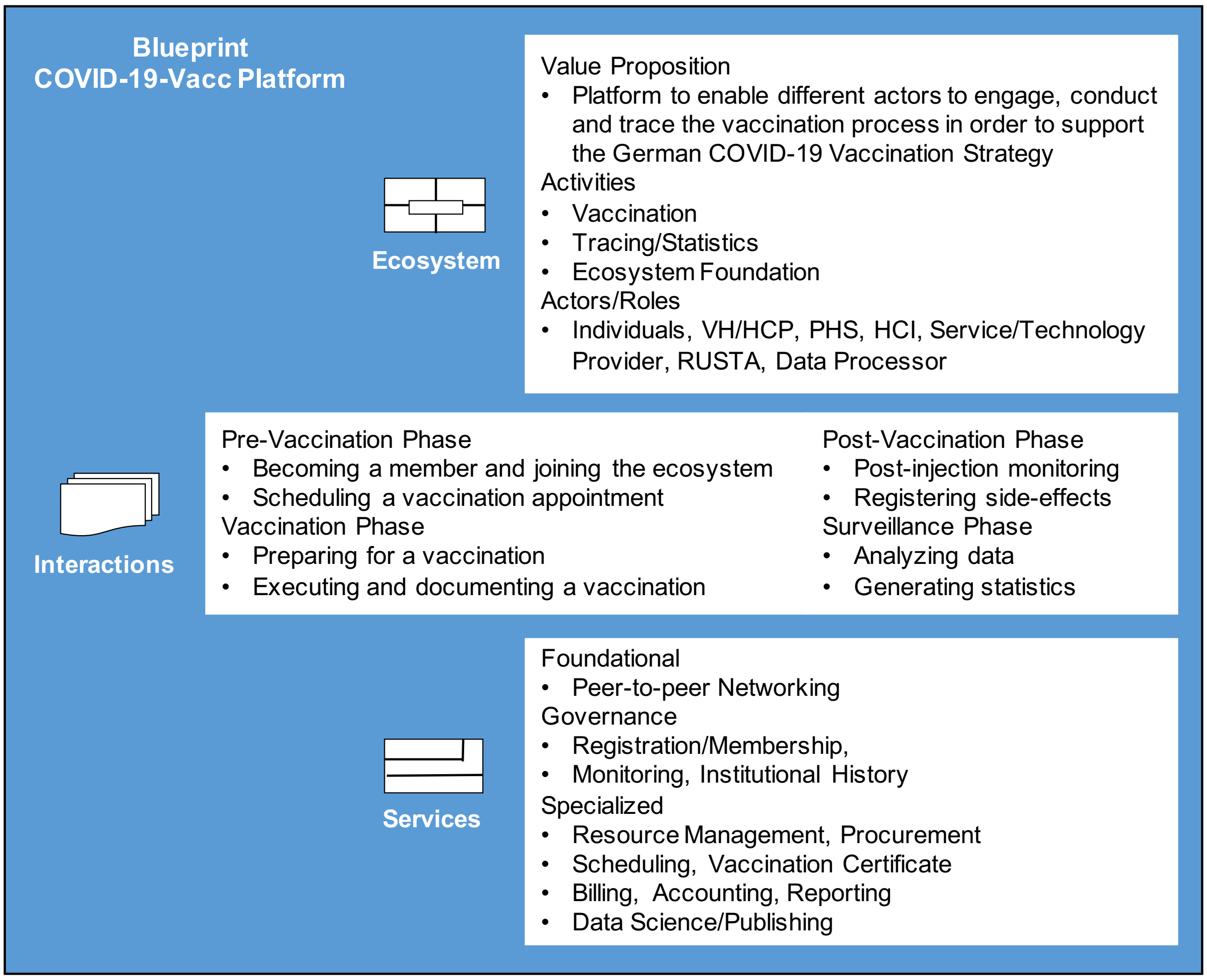


(b) Design steps and related activities

Figure 6.15: COVID-19-VACC design process

Concerning the Step 4, when it comes to design and deployment of the infrastructure for COVID-19-Vacc, the architecture for distributed market spaces (cf. Chapter 5) can be used as a starting point for the implementation, and thus for the concrete realization of a underlying COVID-19-Vacc infrastructure as demonstrated on wemarketspace in Section 6.1.3.

The outcomes of each of the conducted design steps are summarized in Figure 6.16 and explained in the following.



Value Proposition
- Platform to enable different actors to engage, conduct and trace the vaccination process in order to support the German COVID-19 Vaccination Strategy

Activities
- Vaccination
- Tracing/Statistics
- Ecosystem Foundation

Actors/Roles
- Individuals, VH/HCP, PHS, HCI, Service/Technology Provider, RUSTA, Data Processor

Pre-Vaccination Phase
- Becoming a member and joining the ecosystem
- Scheduling a vaccination appointment

Vaccination Phase
- Preparing for a vaccination
- Executing and documenting a vaccination

Post-Vaccination Phase
- Post-injection monitoring
- Registering side-effects

Surveillance Phase
- Analyzing data
- Generating statistics

Foundational
- Peer-to-peer Networking

Governance
- Registration/Membership,
- Monitoring, Institutional History

Specialized
- Resource Management, Procurement
- Scheduling, Vaccination Certificate
- Billing, Accounting, Reporting
- Data Science/Publishing

Figure 6.16: Blueprint of COVID-19-Vacc Platform – summarizing the primary outcomes of the modeling steps and activities (cf. Figure 6.15b)

Step 1 Defining Objectives and Outlining Ecosystem

The objectives identified as primary design goals that an open a decentralized COVID-19-Vacc organizational model needs to meet in order to comply with the German “National Vaccination Strategy COVID-19” can be summarized as follows:

Objective 1 – Facilitating decentralization and openness. Due to the federal setup of the German state as an underlying organizational structure, a COVID-19-Vacc needs to follow the principles of decentralized implementation. Such a platform model needs to recognize and cope with local diversity and differences among different federal states without a central instance of control – as all federal states are equal in their rights and responsibilities. At the same time, it needs to provide a basic organizing structure or a minimum viable formation each federal state can join and become a member of at its own pace. Furthermore, such a platform model has to be open to new actors that might want to join further down the road in implementing the vaccination strategy.

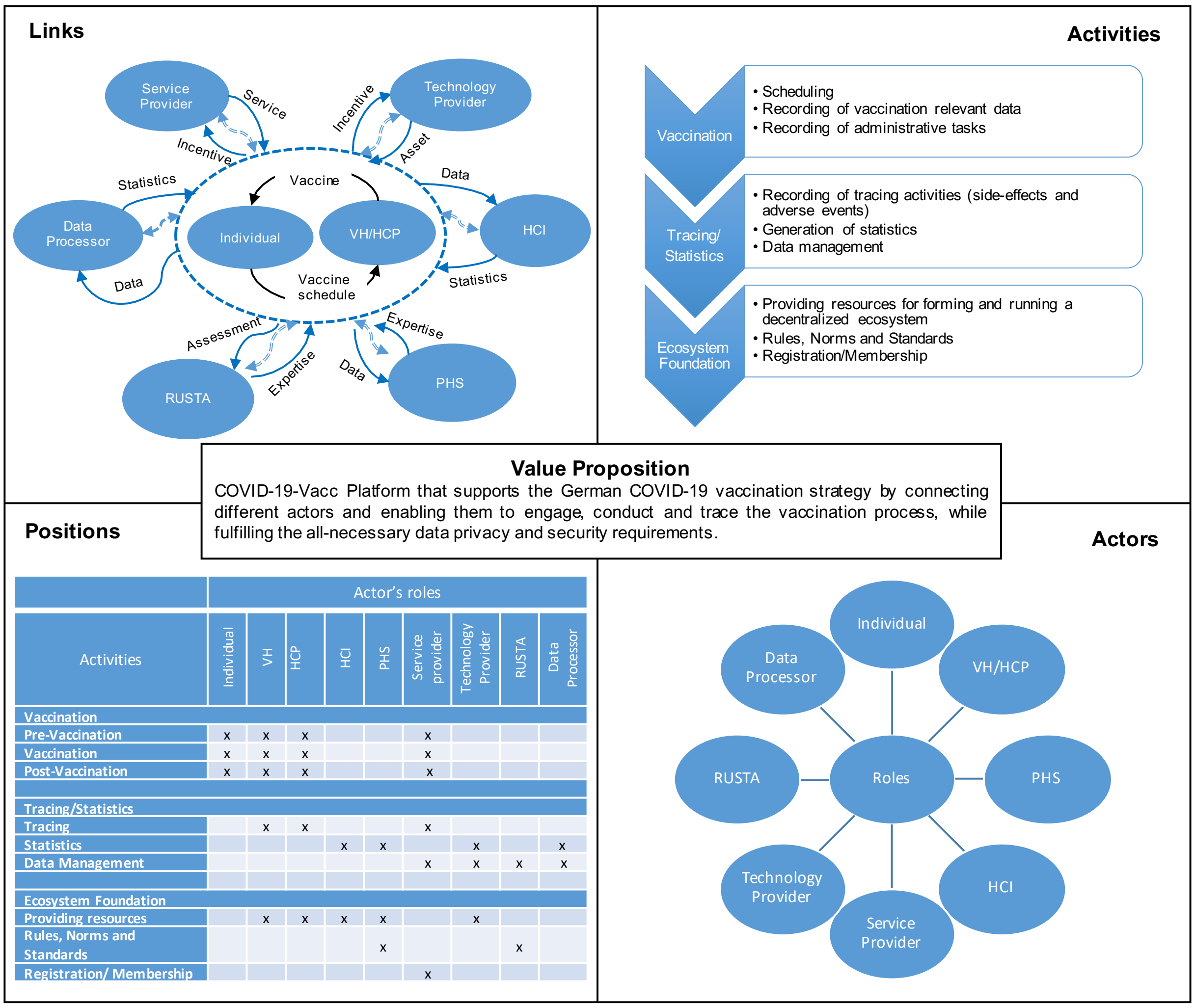
Objective 2 – Enabling the vaccination process. On the operational level, a COVID-19-Vacc needs to provide for the critical functions required for enabling activities related to the four phases of the vaccination process, i.e., the pre-vaccination, vaccination, post-vaccination, and surveillance phases (cf. Section 6.2.1).

Objective 3 – Ensuring information security and data protection during all phases of the vaccination process. All information stored and manipulated by a COVID-19-Vacc and its underlying information systems needs to be aligned with regulations defined by the European General Data Protection Regulation (EU-GDPR) and other data protection laws. Consequently, privacy rights must be ensured and all personal data need to be anonymized, and as such to be used as metadata for statistics and surveillance. In the surveillance phase, personal data should only cease to be anonymous if the person to whom the data belongs experiences side or long-term effects. In practice, this may be a Healthcare Professional (HCP) who looks at the patient's disease symptoms that occurred long after the vaccination. If these symptoms are directly or indirectly linked to the vaccination, it must be possible to trace them back to the vaccination dates.

Objective 4 – Simplicity of use, integration, and management. For a COVID-19-Vacc, existing technologies need to be utilized in a way to implement services that are simple to use, easy to integrate and manage. Therefore, a higher level of automation has to be reached by providing tools and services that support the vaccination process, but also each of the actors involved. For example, these can be well-defined, context-aware service interfaces for scheduling, identifying vaccinated persons (e.g., vaccine certificate), statistics, and dynamic service composition and analytics possibilities.

Objective 5 – Easy to set up, robust, scalable, and modifiable. A COVID-19-Vacc and its underlying software systems and well as its technical infrastructure need to be easy to set up on the deployment level and be robust regardless of its decentralization. Besides, it must be scalable vertically but also horizontally. Vertically in the sense of being able to integrate all sixteen federal states regardless of their intensity of use, without provoking failures, and horizontally to incorporate a broader range and variety of involved actors, which might take different roles within the platform's organizational structure. Moreover, a COVID-19-Vacc must be modifiable in its structure and evolve in its mode of operation to cope with possible changes made to the vaccination strategy as a result of changes in infection numbers.

Based on these primary design objectives, the COVID-19-Vacc ecosystem was outlined and summarized in Figure 6.17.


Links
Service Provider
Technology Provider
Service
Incentive
Incentive
Asset
Statistics
Vaccine
Data
Data Processor
Individual
VH/HCP
HCI
Data
Vaccine schedule
Statistics
Assessment
Expertise
Expertise
Data
RUSTA
PHS
Activities
Vaccination
• Scheduling
• Recording of vaccination relevant data
• Recording of administrative tasks
Tracing/ Statistics
• Recording of tracing activities (side-effects and adverse events)
• Generation of statistics
• Data management
Ecosystem Foundation
• Providing resources for forming and running a decentralized ecosystem
• Rules, Norms and Standards
• Registration/Membership
Value Proposition
COVID-19-Vacc Platform that supports the German COVID-19 vaccination strategy by connecting different actors and enabling them to engage, conduct and trace the vaccination process, while fulfilling the all-necessary data privacy and security requirements.
Positions
Actors
Individual
Data Processor
VH/HCP
RUSTA
Roles
PHS
Technology Provider
HCI
Service Provider

	Actor's roles								
Activities	Individual	VH	HCP	HCI	PHS	Service provider	Technology Provider	RUSTA	Data Processor
Vaccination									
Pre-Vaccination	x	x	x			x			
Vaccination	x	x	x			x			
Post-Vaccination	x	x	x			x			
Tracing/Statistics									
Tracing		x	x			x			
Statistics				x	x		x		x
Data Management						x	x	x	x
Ecosystem Foundation									
Providing resources		x	x	x	x		x		
Rules, Norms and Standards					x			x	
Registration/ Membership						x			

Figure 6.17: Ecosystem model of the COVID-19-Vacc

Value Proposition

The value proposition of the COVID-19-Vacc is formulated as an open, decentralized platform that supports the German COVID-19 vaccination strategy by connecting different involved parties and enabling them to engage, conduct and trace the vaccination process while fulfilling the all-necessary data privacy and security requirements.

Activities

Activities necessary to be conducted by the actors involved in COVID-19-Vacc are those defined by the extended vaccination process, which can be divided up into three main groups. As per Figure 6.17 (top right) these are activity flows related to the vaccination process (i.e., scheduling and recording of vaccination-related data and administrative tasks), and the surveillance (i.e., monitoring, knowledge dissemination, and data management) as well as to the ecosystem foundation (i.e., forming and runing a decentralized ecosystem under consideration of rules, norms, and standards).

Actors and Roles

The actors expected to participate and accomplish these activities in the COVID-19-Vacc ecosystem are individuals, organizations, institutions, and third parties. As depicted in Figure 6.17 (bottom right), these are (at least) nine roles actors might take on.

- Individual – The individual can be a natural person or a legal entity. In our application example, the individual is a person who wants to be vaccinated as part of his or her vaccination journey, either once or several times. Therefore it is important that the active use of the platform is optional for the individual, so that the individual can still get vaccinated without the use of a platform interface. Nevertheless, the individual's data will be processed by the platform. Legal entities, i.e., companies, are included here since companies as a whole may also become engaged in vaccinations in a broader version of the vaccination journey.
- VH (Vaccination Hub)/HCP (Healthcare Professional) – The Vaccination Hub primarily represents the vaccination sites needed to mass-vaccinate the population. Healthcare professionals can also be doctors' offices and hospitals. This overarching grouping of the vaccination infrastructure serves in particular to facilitate the scheduling of appointments and the supply of vaccines to the population.
- HCI (Healthcare Intelligence) – In our application example, healthcare intelligence institutions are the evaluating and advisory bodies that also make recommendations for action. In the example of the Federal Republic of Germany, this can be the Robert Koch Institute (RKI), the Paul Ehrlich Institute (PEI) or others. In our application example, these institutes are sinks for information on infection figures and at the same time sources of expertise, such as recommendations for action to the federal government and subordinate agencies. The HCIs also provide forecasts and produce statistics. They, themselves, are sources of information for the broad media and providers of information to the population.
- PHS (Public Health Services) – Public health services are governmental authorities, such as the Federal Ministry of Health, where all federal decisions are made. These public services exist at the state level as a health center or state department and at the municipal level as a health department. For example, the state ministries are responsible for scheduling vaccination appointments and organizing vaccinations in nursing homes.
- Service Provider – Service Providers provide all services related to the vaccination journey. Examples include vaccine manufacturers, hygiene product manufacturers, suppliers, logistics, and waste management. Service Providers are grouped into one role here, but can take on additional roles in the platform infrastructure.
- Technology Provider – The technolgy provider provides all services that ensure the technical implementation of the approach described here. The role of the technology provider is not a service role, since technology must be standardized to the extent that changes in the platform environment can be adapted quickly by all other roles. If the technology provider were part of the service providers, it would also be subject to the changes and would not be able to deliver a solution

that is independent of these changes. The technology provider works closely with the Rules and Standards Agency (RUSTA).

- Rules and Standards Agency (RUSTA) – The Rules and Standards Agency represents all rules, standards and norms that apply in the platform environment. All roles, without exception, must adhere to the specifications of RUSTA. Changes to the standards are also evaluated and implemented by RUSTA. It is the regulation owner of the platform. The regulations refer in particular to the correct implementation of laws and regulations (e.g., European General Data Protection Regulation (EU-GDPR)) in the respective federal state in which the platform operates and the technological standards implemented by the technology provider.
- Data Processor – The data processors are summary roles around vaccinations and the vaccination journey. These are, e.g., science, statistics, media without direct source reference, and others. From the perspective of the decentralized platform, the data processors are exclusively data sinks and do not provide any data back to the platform and its roles.

Positions

For the stated value proposition to realize, the definition of the necessary activities and the identification of actors who need to undertake these activities, as well as the assignment of roles, are required but not sufficient. The different actors in the COVID-19-Vacc ecosystem need to align around the activities and take on a dedicated position in the overall value creation in order to create value. As illustrated in Figure 6.17 (bottom left), positions provide an overview of where in the flow of activities the identified actors need to be located. Single roles can contribute to several activities, and specific activities might require several roles to engage. For example, to enable vaccination-related activities, individuals, VH/HCP, and service providers need to align, or to support data management, actors taking the role of a service and technology provider, RUSTA, and the data processor are considered essential.

Links

Links illustrate how the actors of the COVID-19-Vacc ecosystem need to interact, and specify their connections. Figure 6.17 (upper left) visualizes these connections (i.e., links) in the form of a flow diagram, which outlines the overall pattern of exchanges within the COVID-19-Vacc ecosystem. The focus lies on the vaccination process, both for individual and VH (Vaccination Hub) or HCP (Healthcare Professional), and the visualization of the most important interactions with other roles and resulting exchanges. Thereby the nodes represent actors performing a particular role, and the arrows the essential interactions indicating the “value exchanged” between these roles. Solid lines denote the so-called “tangible” exchanges, such as vaccines in the case of an individual or data in regards to a data processor, HCI (Healthcare Intelligence), and PHS (Public Health Services). Dashed lines indicate an additional value that is considered “intangible” like feedback or information. Such additional exchanges are considered valuable as they can increase the overall value of the COVID-19-Vacc for its current and future actors, and thus increase trust and ensure the success of the underlying vaccination strategy.

Step 2 Defining COVID-19-Vacc:Interactions

The interaction view specifies the core interaction processes among actors taking on different roles at the operational level. In this case, it concretizes the previously described phases pre-vaccination, vaccination, post-vaccination, and surveillance). From a high-level perspective, these processes and the corresponding actors are illustrated as CMMN diagram in Figure 6.18.

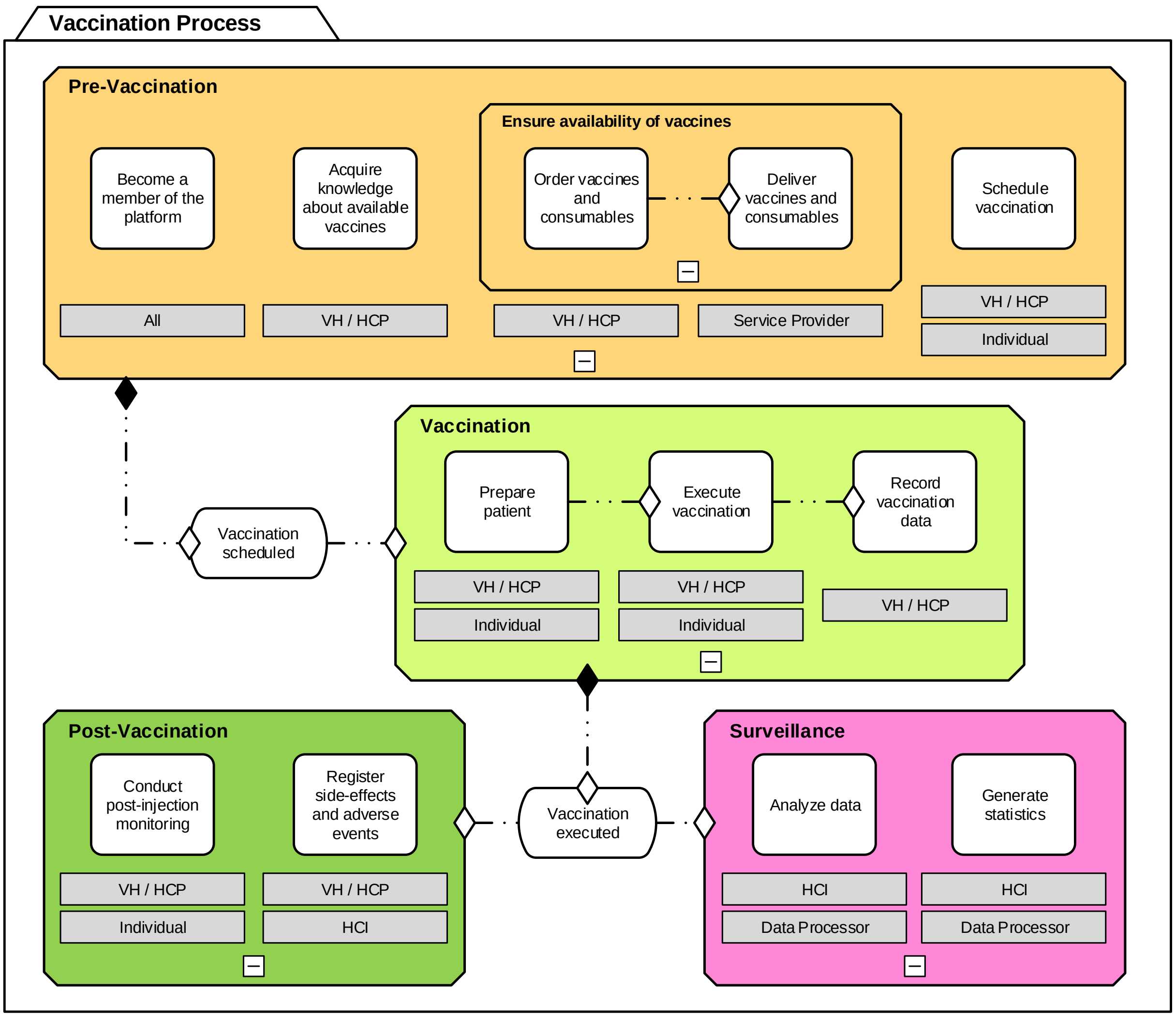


Figure 6.18: High-level overview of interactions supported by the COVID-19-Vacc

The core interaction processes can be specified and, in practice, take on many different forms. The COVID-19-Vacc supports each of these processes by connecting actors and providing common resources. For example, one of many possible concretizations of the process of "generating statistics" in the surveillance phase is the analysis of the effectiveness of measures against COVID-19. For instance, data available on the platform enables data scientists to analyze data about severe courses of the disease in comparison with specific areas, family backgrounds, working from home vs. coming to the office and so on. Through anonymization the platform enables such statistical evaluations and publication of data while protecting data privacy.

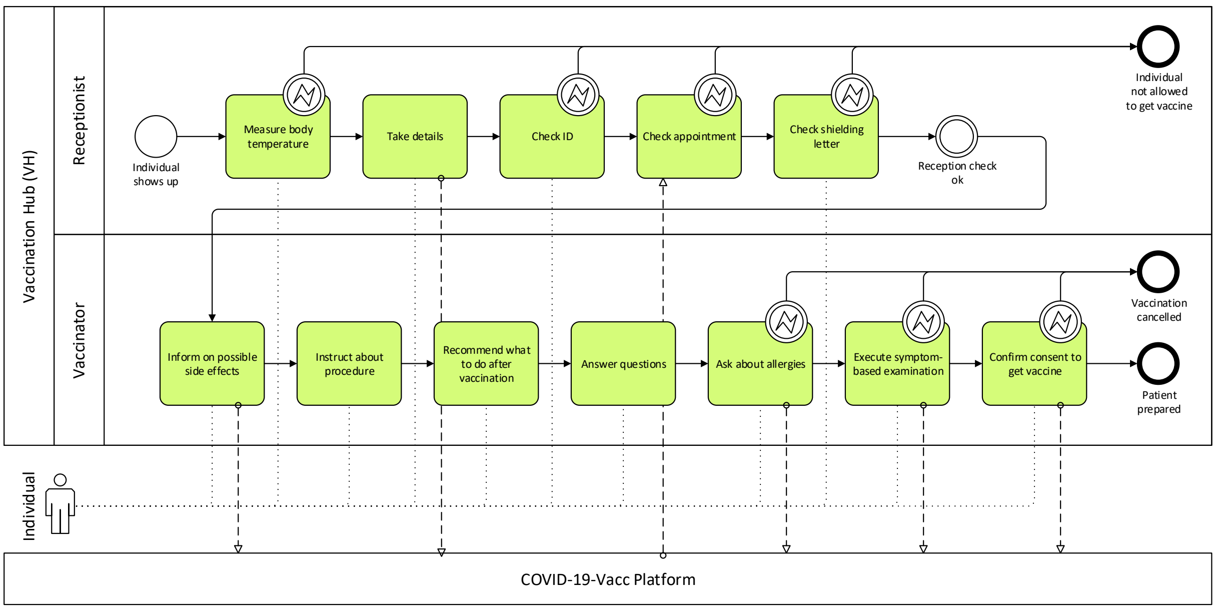


Figure 6.19: Exemplary patient preparation process from a vaccination hub (VH) perspective

Another example of a concretized core interaction process is illustrated as BPMN diagram in Figure 6.19. The figure illustrates an example of the “patient preparation” process, which is part of the vaccination phase and comprises activities, such as measuring the individual’s body temperature and checking the ID before he or she gets inoculated. The process is shown from a VH (Vaccination Hub) perspective, which means that only activities assigned to participant roles on the side of the VH, in this case a receptionist and a vaccinator, are modeled. Two more process participants, namely the individual and the COVID-19-Vacc, are graphically connected to the activities where they are involved. The modeled process starts when an individual shows up at the VH and ends when the individual is prepared for vaccination. However, exceptional situations, for example when the individual has a raised temperature, can lead to the process being terminated, with the individual not being allowed to get inoculated or the vaccination being called off in the end. The COVID-19-Vac is involved throughout the process, for example by providing information about existing appointments and by allowing to document identity checks, the information about e.g., possible side effects or allergies.

Step 3 Defining COVID-19-Vacc:Services

Based on the core interactions, a set of services is identified. As presented in Figure 6.18, these are foundational services necessary for peer-to-peer networking followed by a basic set of governance services for the facilitation of essential self-governance mechanisms of the COVID-19-Vacc ecosystem. In addition to services membership and institutional history to ensure transparency and trust across the ecosystem, two further services are considered necessary, namely monitoring and anonymization. A monitoring service is needed to comply with rules and norms defined by the RUSTA (see Figure 6.17). On the other hand, an anonymization service is required to ensure the proper handling of health data, which is considered a distinct category of personal data, the handling of which is principally prohibited (see Art. 9 EU-GDPR).

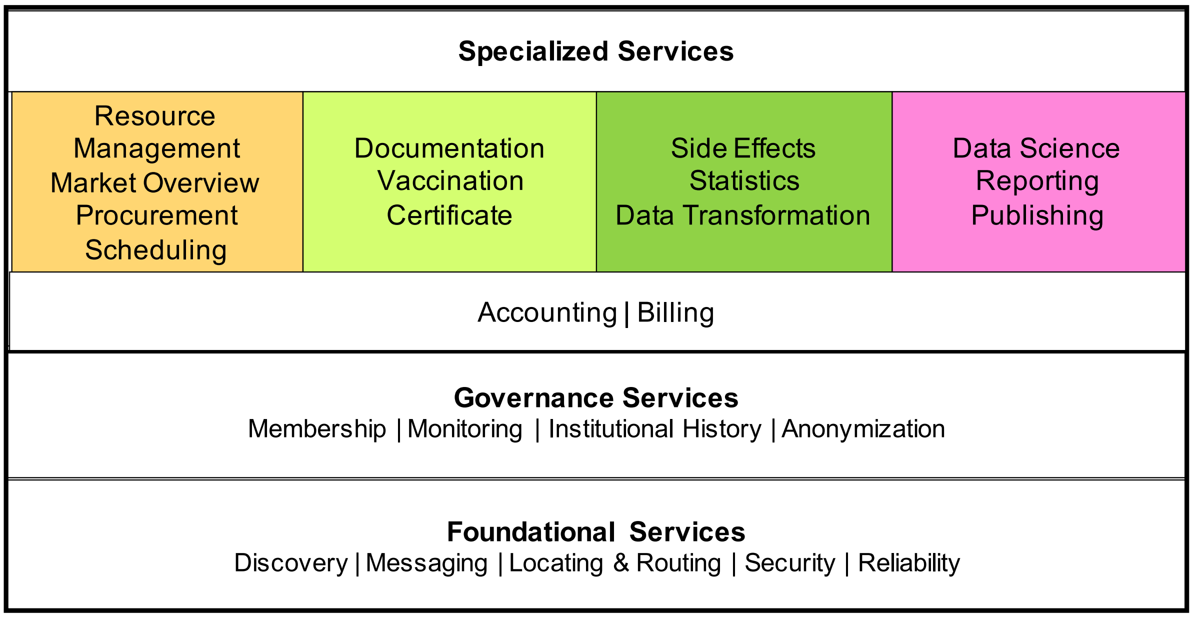

Specialized Services
Resource Management Market Overview Procurement Scheduling
Documentation Vaccination Certificate
Side Effects Statistics Data Transformation
Data Science Reporting Publishing
Accounting | Billing
Governance Services
Membership | Monitoring | Institutional History | Anonymization
Foundational Services
Discovery | Messaging | Locating & Routing | Security | Reliability

Figure 6.20: Service stack of the COVID-19-Vacc

Specialized services (on the top of Figure 6.18) are generally required to support the activities related to the vaccination process. Most of them are needed in one of the four phases, except accounting and billing services, which are considered to be across phases. Identified specialized services are, therefore, the following:

- Resource management service – that supports the central administration, that is warehouse and personnel management depending on scheduled appointments.
- Market overview service – that enables effective resource allocation like vaccines and accompanying material.
- Procurement service – that support ordering and delivery of vaccine and all kinds of consumables such as masks or syringes.
- Scheduling service – that enables individuals to make appointments at VHs or HCPs (Vaccination Hubs or Healthcare Professionals).
- Documentation service – that allows identity check, explanation about possible side effects, privacy policy, and other relevant information.
- Vaccination services – that supports the documentation of the actual vaccination with the unique code of the vaccine batch and personal data of the vaccinated individual.

- Certification service – that supports the generation of a printed and a digital vaccination certificate.
- Side effects service – that supports the documenting of the side effects of the vaccination.
- Statistics service – that enables VHs or HCPs to generate internal statistics about their accomplished tasks.
- Data transmission service – that passes all relevant data and statistics to authorized participants.
- Data Science service – that transforms the plain data to knowledge, which other services can use in the surveillance phase.
- Reporting service – that generates regular reports for knowledge dissemination to subscribers.
- Publishing service – that facilitates publishing work results in scientific publications, press releases, or leaflets for public information.
- Billing service – that allows the billing of different kinds of goods and services among the different participants. For example, a service provider (i.e., the manufacturer) charges the VH (Vaccination Hub) for the delivery of vaccination doses and consumables, or the VH charges another service provider (i.e., a health insurance company) for vaccinating one of their customers.
- Accounting service – that supports the general bookkeeping of all participants involved in the vaccination process.

While the foundational and governance services are more or less relevant for every participant of the ecosystem, most of the specialized services are considered only valid for a subset of those participants. An overview of the specialized services and relevant participants are listed in Table 6.1. In the pre-vaccination phase, resource management is the key for VHs or HCPs (Vaccination Hubs or Healthcare Professionals), because they need to keep track of their warehouses and other resources, like personnel. In addition, suppliers, which are one kind of special providers, need the service to assign their limited goods to their customers. Different service providers can potentially participate in this service, if one of the VHs/HCPs decides to outsource their resource management. The market overview and procurement services are used by the same participants. The scheduling service is always used by the combination of an individual and a VH or HCP, respectively.

However, the service stack of the COVID-19-Vacc can also include a service provider that takes care of the scheduling. All services in the vaccination phase naturally involve the individual to be inoculated, and a VH/HCP. The services in the post-vaccination phase involve different actors: The side effects service involves a vaccinated individual who reports a side effect to a VH or HCP. The statistics service receives information from the VHS, HCPs, and service providers. Thereafter, HCI (Healthcare Intelligence) institutions, PHS (Public Health Services), and other data processors use this data for statistics. However, VHs and HCPs may also create their own statistics.

Table 6.1: List of specialized services and involved roles of the COVID-19-Vacc service stack

Specialized Service	**Involved Roles**
Resource management	VH/HCP, Service Provider
Market overview	VH/HCP, Service Provider
Procurement	VH/HCP, Service Provider
Scheduling Documentation	VH/HCP, Individual Service Provider Individual, VH/HCP
Vaccination	Individual, VH/HCP
Certification	Individual, VH/HCP
Side effects	Individual, VH/HCP
Statistics	VH/HCP, Service Provider, HCI, PHS, Data Processor
Data transmission	VH/HCP, Data Processors, HCI
Data science	Data Processors, HCI
Reporting	Data Processors, HCI
Publishing	Data Processors, HCI
Billing	All
Accounting	All, except Individual

Finally, VHs and data processors use the data transmission service to send their data to HCI. In the surveillance phase, only data processors and HCI are involved in all services. Across all phases, actors need a billing service, as described above. The accounting service is also necessary for most participants, but not for the individual, because, unlike the others, it has no obligation to do accounting. Specialized services may vary as the objectives of the COVID-19-Vacc unfold.

6.3 Discussion

The outcomes of the two applications presented above suggest that the reference model for distributed market areas fulfills the most important common characteristics of a reference model: best practices, universal applicability, and reusability. As shown in the wemarket.space and COVID-19-VACC projects, the proposed reference model can serve as best practice to conceptualize and implement distributed market spaces for a specific application context in the post-platform economy. A more detailed discussion of how it can be used as an instrument for analyzing, designing, and implementing a distributed market space instance in a specific application context is provided in Section 6.3.1.

Furthermore, the previously presented outcomes show that the proposed reference model is universally applicable for various post-platform scenarios. The Smart City environment and the wemarket.space project represents a post-platform scenario for two reasons: Firstly, wemarket.space recognizes consumers and providers as the primary drivers of economic exchange in a smart city as a service ecosystem. It focuses on seamless consumer experiences that enable anyone and anything connected to the internet to contribute and meet such personalized needs. Secondly, wemarket.space regards all participants in their rights and obligations as the same, as they can get involved directly without intermediaries and the associated restrictions. They are constitutive components due to the intention to participate in market exchange and provide the underlying infrastructure and market mechanisms. Although the application domain considers the context of a smart city, the same applies to more significant areas such as smart regions and ultimately non-geographic areas such as open market spaces on the internet scale. This also holds true for the health care scenario, pandemic management, and COVID-19 Vacc. Due to the federal structure of the German state as the underlying organizational structure, a COVID-19 Vacc ecosystem must follow the principles of decentralized implementation to recognize and cope with local diversity and differences between different federal states, since all federal states have their rights and responsibilities. It must also be open to new actors who come in later and integrate and allow them to carry out activities seamlessly.

Moreover, the second application scenario demonstrates the reusability and expandability of the proposed reference model. It shows how an instance of the COVID-19-Vacc environment as an open, decentralized platform model, which can be blueprinted using the adjusted and adapted reference model for distributed market spaces. The adjustment and adaptation were necessary to meet the specific requirements of the pandemic scenario, which are defined in the national COVID-19 strategy and the EU-GDPR (cf. Section 6.2). However, the most significant adjustment and adaptation looked at the underlying phase model to include the phases of the vaccination process (pre-vaccination, vaccination, post-vaccination, and surveillance) instead of the market transaction phases. The results of the blueprinting process show that these models differ from the wemarket.space models from the first application scenario. The main differences result from the adapted reference model and the specific activities, actors, and processes of pandemic management that derive from the contextual requirements of the application context. In that case, the contextual requirements are defined by the objectives of the national COVID-19 strategy and the EU-GDPR constraints. As a result, the service stack of the COVID-19-Vacc differs significantly from the service stack of wemarket.space. One reason is the specific requirements for specialized services resulting from different activity sequences of the underlying vaccination process covered by the COVID-19 platform. Another one refers to additional requirements for governance services resulting from the EU-GDPR restrictions, such as personal health data and the alleged anonymization. Consequently, the reference model for distributed market spaces can be understood as a blueprint for designing and developing decentralized, transaction-oriented environments and is thus considered reusable for various post-platform scenarios.

In addition to these general characteristics of best practice, applicability, and reusability, the reference model for distributed market spaces is required to fulfill specific qualities, which are defined by the design goals stated in Table 4.1. The following sections discuss each of these design goals (RM_DG_1, RM_DG_2 and RM_DG_3). The focus of the discussion is on how the proposed reference model meets the design goal, and how users could benefit from using the reference model when it comes to establishing value ecosystems on their own to engage in post-platform scenarios directly and reliably.

Aligning with principles for appropriate reference modeling (RM_DG_1)

The proposed reference model for distributed market spaces corresponds to the principles of appropriate reference modeling. As formulated in Chapter 4, this design goal is represented by technical model features, language features, and the model's understandability. While the first two assess the general validity of the proposed reference model, the understandability depends heavily on the users, their ability, and their willingness to deal with the reference model and use it effectively.

Technical model features

The technical model features of the reference model for distributed market spaces is described by the formal correctness, architecture, and adaptability, which determine its abstraction level. The proposed reference model is considered formally consistent with the metamodel (MRM [Sch97], [SL98]), which is adjusted and extended to meet the requirements deriving from the domain of the distributed market spaces. By applying the construction principles of instantiation and specialization (cf. Section 4.2), the MRM is used as a meta-model to instantiate the core elements of the resulting model and use that as the basuis for further construction activities. In this way, consistency with the employed meta-model is seen as ensured. The same holds true for the completeness of the proposed reference model, since it is, to a large extent, structurally and behaviorally valid for the requirements of the distributed market spaces as the purpose domain. The structural and behavioral completeness is given through the adjustments and modifications made to meet the specificities of the purpose domain. In concrete terms, the meta-model's vertical dimension (i.e., views) is modified by replacing the business view with the ecosystem view, the horizontal dimension (i.e., phases) by extending and rearranging existing and adding a new phase to integrate activities and processes relevant for the transaction of complex products. And finally, an entirely new dimension is added to integrate aspects belonging to different lifecycle stages and thus meet the design goal(RM_DG_3), as discussed in Section 6.3.

Based on these design decisions, an extended model architecture is created that not only includes multi-views and multi-phases, but is multidimensional. Furthermore, the applied principles of abstraction and encapsulation on the architectural level contribute to the adaptability of the proposed reference model. As the applications in the context of Smart City and COVID-19-Vacc show, the architecture of the reference model allows adjustments and even significant changes of invariant elements, such as the underlying phase model. Invariant elements do not denote the activities within a phase, but the respective phases with a set of entirely different activities. For modeling the COVID-19-Vacc (cf. Section 6.2.2), the phase model was changed to meet specific contextual requirements of the vaccination process. These changes have resulted in inheritance in the variant parts, but they do not cause side effects on other invariant parts of the model. For example, the service stack has changed due to new identified specialized and additional governance services. Still, the architecture of the service stack as a structured collection of the foundational, governance, and specialized services remained unchanged. The same applies to the structure of the ecosystem model as an invariant part, although different activities, roles, and interactions between them are defined. In this way, and given the outcomes of the two conducted application scenarios, the adaptability of the reference model for distributed market spaces is considered to be sufficient. However, further applications in different application domains are recommended for a final assessment of the adaptability of the proposed reference model.

Language features

The proposed model for distributed market spaces uses the notation of the leveraged meta-model, which is represented by boxes and lines. However, for the representation of the interaction view, it uses the standardized notation BPMN (BPMN 2.0 [OMG15]. BPMN is common for describing business processes. Moreover, for the description of the underlying software architecture, the notation UML (UML 2.5[Obj15]) is used to describe statical elements of the functional structure, its components, and interfaces they expose, as well as to describe the dynamic behavior of these components. Both BPMN and UML are inherently highly standardized but also extendable languages. This supports the adaptation of the proposed reference model as demonstrated with adapted processes of the interaction view on example COVID-19-Vacc (cf. Section 6.2.2). The design decision to rely on the inherited "boxes and lines" and the newly added notations BPMN and URL foster the integrity and the reference model's consistency. However, they also bear some issues due to the lack of clear mapping rules for creating the model. Such a high degree of freedom of choice for the model's user (i.e., modeler) might result in ambiguous models with different interpretation options and thus a need for additional communication among the involved users.

Understandability

The understandability of the proposed reference model is primarily given due its clear, compact structure, which is underpinned by the separation of concerns into three dimensions, four views, and five phases. Despite the clear separation of concerns, views and phases are linked within one dimension and thus form a comprehensive, multi-dimensional, multi-view reference model. For different stakeholders, that is for model users, such a structure offers a ”big picture” of the concerns and activities that have to be taken into account during the modeling process. And on the other, it enables different stakeholders to carry out different modeling steps related to different views and phases. This is important because stakeholders are considered different model's users, and due to their competencies and preferences are usually involved in certain modeling activities. For example, business and process competencies are relevant for modeling on the strategic level (ecosystem and interaction view), or software engineering competencies when implementing the defined processes on the operational level (service and infrastructure view).

Another aspect relevant for the usability of the proposed reference model is the use of the familiar terminology and modeling language used to describe the model's dimensions, views, and phases. The terminology used is in line with the common terminology used in the domain of platform economy and generally commercial exchange over the Internet so that users might immediately start using the model. Furthermore, the usability of the model is also supported by the simple and arguably intuitive graphical representation used to visualize different dimensions, views, and phases. For example, a simple visual representation is used to visualize the ecosystem view (cf. Figure X) and the ecosystem structure. The elements value proposition, actors, positions, and roles are arranged in a clockwise square. The sequence suggests the dependencies between the elements and the steps in which the analysis of the respective elements should be carried out. This representation follows the visualization logic of the well-known Business Model Canvas by Osterwalder et al. [Ost+11]. It might correspond to concepts the prospective model's users are familiar with and thus contribute to the comprehensibility of the reference model and increase the user's willingness to use it.

Guiding the Analysis, Design, and Implementation of an Instance of Distributed Market Space (RM_DG_2)

The results of the two applications presented above suggest that the reference model for distributed market spaces meets the design goal of serving as a guide for the analysis, design, and implementation of scenarios in the post-platform economy. In following each of these three aspects, analysis, design and implementation are discussed.

Analysis

The proposed reference model supports a structured analysis of aspects of the strategic and operational levels of a decentralized and self-organized online structure. As an analysis instrument, it separates concerns of these two levels guiding the analysis through three dimension phases, views, and stages. Each of these dimensions has a different focus and compounds of different elements that need to be analyzed separately in order to gain a comprehensive overview. Therefore, the proposed framework assists in gaining a deeper understanding of relevant entities, elements, and relationships between them, which facilitates defining the project scope and resulting design requirements. Furthermore, it supports an early estimation of competencies, considered essential for design activities, and, eventually, implementation (i.e., competencies necessary for modeling interaction processes (interaction view) or IT expertise inherent for activities associated with services and infrastructure). In the wemarket.space and COVID-19-Vacc projects, these competencies were assumed to exist, but this is not generally the case. Therefore, training the project team to apply the reference model for distributed market spaces is highly recommended for the team to use it adequately and beneficially.

Design

As a design instrument, the proposed reference model structures the design process by providing a set of four steps that are organized cyclically in order to build a wheel model. This leads the process from modeling an ecosystem structure over the core interactions and services to its prototypical implementation. Each step relies on the inputs of the previous one, and each step produces a clearly defined outcome in the form of derived models. Together these four models compose a blueprint that blends the results of different design activities on the one side and integrates the application context on the other. This ensures that all models are derived in context and serve the defined value proposition. This work presented only one iteration of the blueprinted models in both application scenarios, as the focus was to implement an instance of wemarket.space and the COVID-19-Vacc. These instances should be seen as one of the first iterations of the design process. As to the applied wheel model (cf. Figure 6.2b), this first instance should be seen as the test system required to test the blueprinted models and underlying design assumptions rather than a final sample of the system.

Implementation

Most examined reference models and approaches (cf. Section 4.2) focus solely on modeling activities, leaving the prototyping to the software engineers and developers responsible for the implementation. The proposed model goes a step further, introducing an accompanying architecture for distributed market spaces as the possible implementation of the infrastructure view. Hence it supports the implementation of a reference DMS on the operational level. This implies two further advantages of the proposed reference model. Firstly, it speeds up prototyping the blueprinted models, giving a detailed description of the underlying software system. Due to the specified functional and information structure of the system, the prototyping activities can focus on the realization of the proposed solution and thus lead to more reliable results in a shorter period. For the prototyping of the wemarket project (cf. Section 4.8), Web technologies have been used. However, technology choice is subject to a concrete implementation for a particular application context and depends on the team's competencies, as mentioned before. Secondly, it enables an early evaluation of the design hypothesis and associated design decisions. The blueprinted structure incorporates design decisions based on the understanding and assumption about the application context and associated requirements. Since these assumptions might not be complete or even valid, the prototyped system will realistically validate the outcomes of the design activities. As a result, possible deficiencies are identified, and modifications are defined regarding the ecosystem structure, interactions, or service stack. Such a feedback loop allows the blueprint to undergo the necessary iterations to arrive at the so-called "minimum viable product" state. That means a minimum viable system is necessary to operate in order to for the defined value proposition to be released.

The proposed reference model concentrates on different aspects of distributed market spaces and their inner workings on the strategic and operational level, which are relevant for market transactions for complex products, and how their instances might unfold during different life stages. However, the proposed reference model omits aspects related to regulatory affairs. These are particularly important for cross-border market transactions (i.e., when providers from different countries are involved). Currently, such cross-country regulations, especially in online trading, differ from country to country. Alternatively, from region to region, or are even part of current political considerations as is the case with the Single Market, Initiative of the European Commission [Com15], just to mention an example. Therefore, such regulatory constraints are to be considered as part of the contextual requirements so that they can flow into the design decisions. Further regulations such as EU-GDPR, which are considered essential contextual requirements as shown in the second application scenario on example COVID-19-Vacc, go in a similar direction.

Moreover, the proposed reference model omits concepts that cope with the identity and privacy of users (i.e., actors). It draws on the positive collective motivation. Yet forgery and fraud are undesired practices that any market-oriented or transaction-oriented environment faces to some extent. Different approaches to this issue exist, e.g., NICE [LSB03], and SOLID [Ber17], that can be taken into consideration to prevent or at least mitigate undesired behavior.

Furthermore, some sensibilities and trade-offs (cf. Section 5.7.2) derive from the strictly decentralized technical infrastructure (i.e., software-system for implementing the infrastructure view). These are sensitivities linked to decentralized governance and data management, and are viewed as the main trade-offs to alleviate the adverse effects of the positional power of centrally governed environments. Nevertheless, these are out of this work's scope and are considered subject to future work.

Assisting in further development of an instantiated distributed market space during its lifecycle (RM_DG_3)

The proposed reference model provides assistance in the further development of an instantiated distributed market space during its lifecycle. This is ensured by the third dimension of the model, which covers three stages a distributed market space might undergo during its lifecycle: design, ignition, and maturity. In that way, it also, in its third dimension, follows the same design principles as with views and phases; the separation of concerns, where each of the life stages has different concerns and therefore different priorities, which in turn necessitate different activities, in order to reach the threshold for the next stage.

As with the applications above, the primary concern of the design stage is to blueprint, prototype, and launch an instance of a distributed market space that considers the contextual requirements of the particular application. However, the next two stages concentrate on the further development of such an instance. The ignition stage thus focuses on building the critical mass of participants and clearing frictions and bottlenecks that occur during the early stage of the instantiation. On the other hand, the maturity stage concentrates on the further development and growth of a running instance of the distributed market space. For each of the stages, a set of activities are presented to serve as a guideline on how to approach the development and growth of an instantiated distributed market space. For the users of the model, it provides a framework that lists possible next steps. As presented in Chapter 4 (cf. Table 4.3), for each of the stages, a set of activities are introduced and linked to the separate views. This provides a possible overview of upcoming activities, which is extremely relevant for strategic decisions and running, and operations. In that way, the proposed stage model, as the third dimension of the reference model, facilitates a structured analysis of the additional activities related to the development and growth, and supports modelers in strategic and operational planning. ,

However, the introduced activities are not exhaustive and might change due to unforeseen factors and challenges that might occur during the lifecycle. These factors and challenges can derive from the specific application context and related contextual requirements. Still, they also might have causes lying in the nature of the platform-based ecosystem models. These are, for example, issues like an insufficient level of trust among the ecosystem or competitive threats corresponding to inner- and outside attacks and hostile take-over attempts, as pointed out in Chapter 4. Therefore, future work on the stage model as the third dimension of the reference model might consider these issues and integrate concepts (e.g., business ecosystem health concept [HTV06], ”5E” approach [RR17]) as possible mitigation strategies, including a catalog of measures as a possible line of action against these threats.

6.4 Summary

This chapter applied the reference model for distributed market spaces to two application scenarios; in the context of a Smart City on the project wemarket. space, and in the context of COVID-19 pandemic management, on the project COVID-19-Vacc. Based on these outcomes, it evaluated how the proposed reference model meets its design goals. Therefore, Chapter 6 presented the fourth main contribution of this thesis and thus the primary results of phase 5 of the modeling process (Figure 4.24).

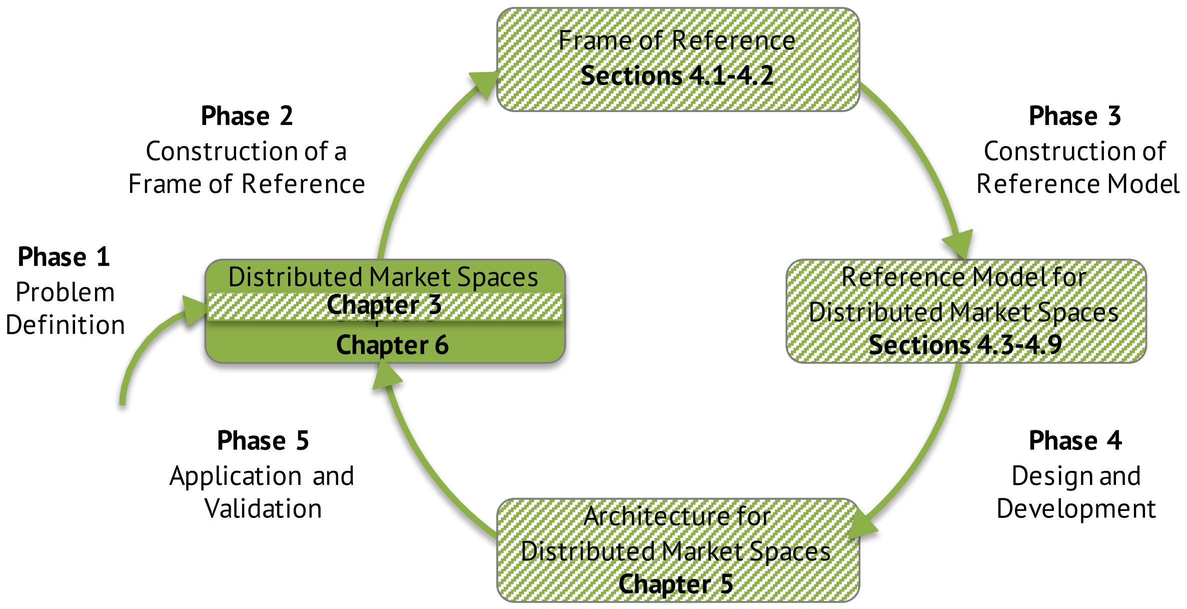

Frame of Reference
Sections 4.1-4.2
Phase 2
Construction of a Frame of Reference
Phase 3
Construction of Reference Model
Phase 1
Problem Definition
Distributed Market Spaces
Chapter 3
Chapter 6
Reference Model for Distributed Market Spaces
Sections 4.3-4.9
Phase 5
Application and Validation
Phase 4
Design and Development
Architecture for Distributed Market Spaces
Chapter 5

Figure 6.21: Contributions of Chapter 6 to the modeling process

The modeling process in both application scenarios took place in four steps, organized as a wheel model. As a result, blueprints for wemarket.space and COVID-19-Vacc were derived. The wemarket.space was launched as a demonstrator in a testbed environment using the actual context of the city of Frankfurt. The demonstrator showed how users could form the wemarket ecosystem and participate in highly personalized city essentials transactions. The outcomes of both applications have shown that the proposed reference model fulfills the primary characteristics of best practice, universal applicability, and reusability. Moreover, its technical model and language feature generally align with principles for appropriate reference modeling. The evaluation results also suggest that it is well understandable for the modeler to serve as guidance for analyzing, designing, and implementing distributed market spaces in a particular domain. Besides, it provides further assistance in developing and growing instantiated distributed market spaces during their lifecycles.

7 | Conclusion and Future Work

The post-platform economy shifts power from platforms to consumers and providers as the primary drivers of market exchanges on the Internet. It recognizes the importance of consumers and their personalized needs; it allows anyone and anything connected to the Internet to contribute to the satisfaction of such personalized demands. Consumers and providers are equal in their rights and obligations in a post-platform economy. They can participate directly in complex product scenarios without intermediaries and are constitutive parts of supporting infrastructures required for an economy that does not rely on centrally orchestrated structures to enable market exchange.

The post-platform economy refers to a set of economic activities facilitated by distributed market spaces. Distributed market spaces are self-organized and strictly decentralized organizing structures that counter the adverse effects of growing platform power and lower transaction costs for complex products while retaining the advantages of the centrally organized platform models.

This thesis proposed a reference model and an accompanying software-system architecture, which together can be used as a guiding framework for the analysis, design, and implementation of distributed market spaces. The primary purpose of this framework is to enable and encourage market participants (consumers and providers) to initiate distributed market spaces on their own and, by doing so, to establish value ecosystems where they can exchange complex products directly and reliably. In that respect, the proposed framework for distributed market spaces can facilitate the emergence of the post-platform economy and more broadly contribute to the current initiatives for the re-decentralization of the Internet as a global market space.

Figure 7.2 summarizes the contributions of this thesis and puts them in relation to the modeling process applied to develop the reference model and architecture for distributed market spaces. Each of these contributions is reviewed in the following:

1. *It introduces distributed market spaces as a new concept for the post-platform economy. It identifies their primary characteristics and overall objectives and defines the scope of distributed market spaces as the domain of interest to reference modeling* [RP15], [PRR16], [RWP17].

   Distributed market spaces are online structures that counter the adverse effects of growing platform power and lower transaction costs for complex products while maintaining the benefits and enabling nature of the centralized platform models. The main characteristics of distributed market spaces are that they are self-organized and strictly decentralized organizing models. The scope of distributed market spaces as a domain of interest to reference modeling is defined by

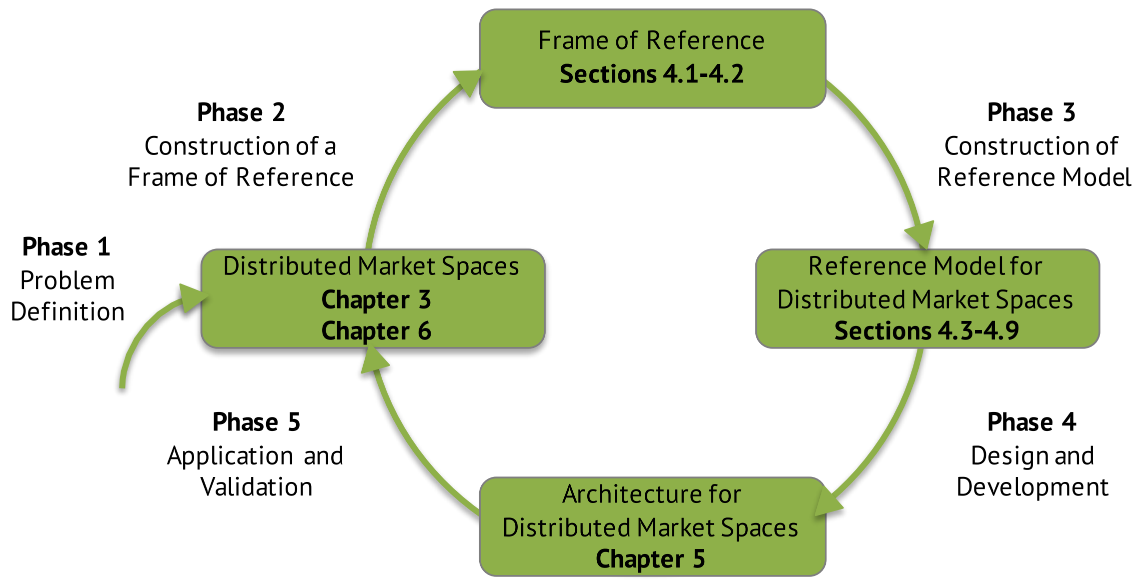

Frame of Reference
Sections 4.1-4.2
Phase 2
Construction of a Frame of Reference
Phase 3
Construction of Reference Model
Phase 1
Problem Definition
Distributed Market Spaces
Chapter 3
Chapter 6
Reference Model for Distributed Market Spaces
Sections 4.3-4.9
Phase 5
Application and Validation
Phase 4
Design and Development
Architecture for Distributed Market Spaces
Chapter 5

Figure 7.1: Contributions of this thesis to the modeling process

its overall objectives. They derived from the BOAT [Gre15] analysis of different business, organization, architecture, and technology-related aspects of complex product scenarios. The identified objectives define distributed market spaces as exchange environments that enable market transactions of complex products directly and reliably, facilitate decentralization, scalability, and openness, and finally support simplicity of use and management (cf. Chapter 3).

2. *It develops a reference model for distributed market spaces for the analysis and design of self-organized structures that lower transaction costs for complex products and facilitate market exchange in a decentralized manner* [RWP17], [RP19a], [RP19b], [RP19c].

   A new reference model for distributed market spaces has been developed. It describes the construct of distributed market spaces (reference DMS), taking three different perspectives: phases, views, and stages. As a multi-dimensional and multi-view reference model, it defines how a reference DMS enables market transactions for complex products (phases), how it works on the strategic and operational level (views), and how its instances might unfold during the different life cycles (stages). The proposed phase model represents a reference DMS as a market-oriented environment, defining how to initiate, arrange, and settle complex products to lower transaction costs and facilitate market exchange. The ecosystem, interaction, service, and infrastructure views define a reference DMS on the strategic and operational levels. The life stages model acknowledges different concerns and related challenges that an instantiated distributed market space might undergo. It proposes a set of activities relevant for the development and growth during different life stages of design, ignition, and maturity (cf. Chapter 4).

3. *It designs, develops and evaluates an* architecture for distributed market spaces *as a strictly decentralized and highly scalable software-system architecture for a concrete implementation of a distributed market space* [PRR16], [Hit+16], [RP19c], [RPR20], [RRP21], [RP19b].

   A novel software-system architecture for distributed market spaces has been designed and developed to serve as a concrete implementation of the infrastructure view of the reference model for distributed market spaces. It represents a blueprint of an information system necessary for implementing a reference DMS on the operational level, thus realizing foundational, governance, and specialized services as defined by the service view of the reference model. It is strictly decentralized and highly scalable, and employs distributed resources to perform the required services, considering the system users as constitutive elements that actively contribute to establishing the system structure and its mechanisms. The proposed architecture has been evaluated by a proof-of-concept implementation and applying ATAM (Architecture Trade-Off Analysis Method) [CKK+03]. The evaluation results showed that the proposed architecture meets the design goals largely; however, sensitivities and trade-offs are identified, and different mitigation approaches and measures are proposed, which are to be considered in the context of a particular application scenario (cf. Chapter 5).

4. *It demonstrates the applicability and feasibility of the proposed reference model for distributed market spaces and the accompanying architecture evaluating them in the context of different application scenarios* [RP19b], [Rad+21].

   The applicability and feasibility of the proposed reference model for distributed market spaces and the accompanying architecture have been demonstrated in two application scenarios; in the context of a Smart City with the project wemarket.space, and in the context of COVID-19 pandemic management, using the project COVID-19-Vacc. The modeling process in both application scenarios took place in four steps, and was organized as a wheel model. As a result, blueprints for wemarket.space and COVID-19-Vacc were derived. The wemarket.space was lounged as a demonstrator in a testbed environment using the actual context of the city of Frankfurt. The demonstrator showed how users could form the wemarket ecosystem and participate in highly personalized city essentials transactions. The outcomes of both applications have demonstrated that the proposed reference model fulfills the primary characteristics of best practice, universal applicability, and reusability. Moreover, its technical model and language feature generally align with principles for appropriate reference modeling. The evaluation results also suggest that it is well understandable for the modeler to serve as guidance for analyzing, designing, and implementing distributed market spaces in a particular application context. Besides, it provides further assistance in developing and growing instantiated distributed market spaces during their lifecycles (cf. Chapter 6).

Future work will concentrate on extending the proposed reference model and architecture for distributed market spaces and improving its elements and components.

The reference model for distributed market spaces can be expanded to include elements that integrate regulatory aspects. They are currently considered as part of the contextual requirements of the respective application scenario. As discussed in Chapter 6, regulatory issues are essential to cross-border market transactions. Today, such transnational regulations differ from country to country. Still, there are initiatives to regulate markets on a geographical basis, as in the case of Single Market [Com15], the part of the European Commission's initiative for Europe. The elements to integrate these aspects affect the dimension view of the reference model. The ecosystem view can be extended to include additional activities and roles that an actor could assume within the ecosystem or interaction, and a service view with various processes and services to support ecosystem view extensions.

Furthermore, the proposed reference model relies on the participants' positive collective motivation to neglect undesirable practices that every transactional environment faces to some extent. Therefore, elements that comply with identity and privacy regulations can be added to mitigate the undesired behavior of involved participants. Specifically, the work of Windrich, Speck, and Gruschka [WSG21] on data protection and Niemand et al. [Nie+15] on the transfer of laws into executable processes go in this direction. Alternatively, it would be conceivable to introduce a new ”Regulations view”, containing general regulations and restrictions relevant to all market transactions. Application-specific regulations and restrictions could still be considered part of the contextual requirements of the respective application scenario. The service view can be expanded to incorporate additional reward mechanisms relevant to peer reputation. This means extending the proposed service stack, especially governance services, for example, with services implementing the concept of social currency [McC14], as discussed in Chapter 5.

Considering the identified sensitivity points of the proposed architecture for distributed market spaces (cf. Chapter 5), new services can be introduced to extend the existing components for decentralized governance and data management. Another aspect related to data management is the chosen RDF data model considered critical to the description of knowledge and ultimately for the market transactions of complex products. The creation and publishing of structured domain knowledge are time-intensive, costly, and require profound domain expertise. Consequently, to launch a distributed market space requires a basic set of domain-related expertise (i.e., ontologies and vocabularies) and high manual engagement of the dedicated role (i.e., knowledge providers). To facilitate this process, we propose an approach to design an ontology to serve as a meta-model to generate user interface models [RRP21]. The user interface models are used to create web applications with dialog-based HTML forms, which are eventually used to populate instances of OWL ontologies. Our meta-model includes several patterns used to generate programming control structures to populate ontology instances. On the one hand, the meta-model describes user interfaces, and on the other hand, it describes the structure of the output ontology instance.

The launched instances of distributed market spaces in two application scenarios prove the concept for the proposed reference model and the accompanying architecture, and show their practical applicability. To be used for real application scenarios, the current demonstrator would have to be extended to implement all services of the proposed service stack. In particular, the governance services must be implemented, the performance of the basic services improved, and tested for security aspects. While the demonstrator has so far only been tested on a local testbed, real-world experiments over the Internet would be required to test the implemented mechanisms regarding the Internet anomalies and the dynamics of the underlying peer-to-peer network.

Finally, the demonstrator currently only supports shaper roles (consumer and provider), as they shape the value proposition and the birth of a distributed market space. The functionality that supports the enabler roles, particularly those of knowledge provider, steward, and reputation bank, must be implemented in order for the demonstrator to support the desired ecosystem structure on the operational level. Thus the value proposition of an end-user-enabled ecosystem for the market exchange of complex products in a decentralized manner would be realized.

As the work has shown, a wide variety of application scenarios are conceivable for using the reference model and the architecture for distributed market spaces. These can be seen in both business and public administration. As a part of increasing digitization, for example, in agricultural, financial, and energy industries, and in the public administration at district, state, federal, and even EU levels in order to sustainably improve public activities.

A | Appendix

A.1 CPR Builder Configuration

Data Description Model and Label Properties for Domain "Babysitting"

```

data_description : "babysitting"
referencedOntologies [
base: "http://xxx.org/ticketingDO/v1#"
tiod: "http://xxx.org/ticketing/v1#"
gr : "http://purl.org/goodrelations/v1#"
xmls: "http://www.w3.org/2001/XMLSchema#"
]
{
group: "data" { swClass="BabysittingRequest"
swIndividual="babysittingrequest"
/*
???
*/
}
{
group: "babysitterdetails"{
name:    {swForIndividual="babysittingrequest"
swProperty="gr:name"
swType="gr:name"

}
prefered:{
restrictedTo="babysitter|nany|agency|all"
initialValue="all"
type="manyOfMany"
swForIndividual="babysittingrequest"
swProperty="tiod:eventcategory"
swType="tiod:eventcategorylist"

/*
swForIndividual="parkingrequest"
swProperty="pod:preferedcategory"
swType="pod:preferedcategorylist"
*/
}

group: "experience" {}
{
age :{ initialValue="none"
restrictedTo="<20|<30|>40|no␣relevance"
}
childagegroup:{ restrictedTo="0-12␣month|1-3␣years|4-12␣years|all␣ages"
    },
availability: { restrictedTo="morning|afternoon|evening|overnight"},

pickuptime : { type="number", unit="Uhr", min="0", max="24"},
duration : { type="number", unit="hrs", min="2", max="24"}
}

}
group: "price_range" {
swClass="gr:UnitPriceSpecification"
swIndividual="hasPriceSpecification"
swForIndividual="parkingrequest"
swProperty="gr:hasPriceSpecification"
/*
???
*/

}
{
maxprice: {
type="number", unit="EUR"
initialValue="1.1"
swForIndividual="hasPriceSpecification",
swProperty="gr:hasMaxCurrencyValue",
swType="xmls:float"
/*
???
*/
}
}
}/*
group :"data" {}
{
summary: { type="snippet"
initialValue="snippets/datamodel.html"
}
}
*/

}
```

Listing A.1: Data description model for the domain "babysitting"

```

### field labels
babysitting = babysitting
babysitting.data = data
babysitting.data.babysitterdetails = About Babysitter
babysitting.data.babysitterdetails.experience = experience
babysitting.data.babysitterdetails.experience.age = age
babysitting.data.babysitterdetails.experience.availability =
    availability
babysitting.data.babysitterdetails.experience.childagegroup =
    childagegroup
babysitting.data.babysitterdetails.experience.duration = duration
babysitting.data.babysitterdetails.experience.pickuptime = pickuptime
babysitting.data.babysitterdetails.name = name
babysitting.data.babysitterdetails.prefered = category
babysitting.data.price_range = salary
babysitting.data.price_range.maxprice = max. price per hour

### value labels for restricted fields
values.babysitting.data.babysitterdetails.experience.age = <20|<30|>40|
    no relevance
####values: <20|<30|>40|no relevance
values.babysitting.data.babysitterdetails.experience.availability =
    morning|afternoon|evening|overnight|all
####values: morning|afternoon|evening|overnight
values.babysitting.data.babysitterdetails.experience.childagegroup =
    0-12 month|1-3 years|4-12 years|all ages
####values: 0-12 month|1-3 years|4-12 years|all ages
values.babysitting.data.babysitterdetails.prefered = babysitter|nany|
    agency|all
####values: babysitter|nany|childcare|all
#values.infos.babysitting.data.babysitterdetails.experience.age = n/a
#values.infos.babysitting.data.babysitterdetails.experience.
    availability = n/a
#values.infos.babysitting.data.babysitterdetails.experience.
    childagegroup = n/a
#values.infos.babysitting.data.babysitterdetails.prefered = n/a
#values.icons.babysitting.data.babysitterdetails.experience.age = n/a
#values.icons.babysitting.data.babysitterdetails.experience.
    availability = n/a
#values.icons.babysitting.data.babysitterdetails.experience.
    childagegroup = n/a
#values.icons.babysitting.data.babysitterdetails.prefered = n/a


### field infos
#info.babysitting = n/a
#info.babysitting.data = n/a
#info.babysitting.data.babysitterdetails = n/a
#info.babysitting.data.babysitterdetails.experience = n/a
#info.babysitting.data.babysitterdetails.experience.age = n/a
#info.babysitting.data.babysitterdetails.experience.availability = n/a
#info.babysitting.data.babysitterdetails.experience.childagegroup = n/a
#info.babysitting.data.babysitterdetails.experience.duration = n/a
```

```
#info.babysitting.data.babysitterdetails.experience.pickuptime = n/a
#info.babysitting.data.babysitterdetails.name = n/a
#info.babysitting.data.babysitterdetails.prefered = n/a
#info.babysitting.data.price_range = n/a
#info.babysitting.data.price_range.maxprice = 10 EUR

### field icons
#icon.babysitting = n/a
#icon.babysitting.data = n/a
#icon.babysitting.data.babysitterdetails = n/a
#icon.babysitting.data.babysitterdetails.experience = n/a
#icon.babysitting.data.babysitterdetails.experience.age = n/a
#icon.babysitting.data.babysitterdetails.experience.availability = n/a
#icon.babysitting.data.babysitterdetails.experience.childagegroup = n/a
#icon.babysitting.data.babysitterdetails.experience.duration = n/a
#icon.babysitting.data.babysitterdetails.experience.pickuptime = n/a
#icon.babysitting.data.babysitterdetails.name = n/a
#icon.babysitting.data.babysitterdetails.prefered = n/a
#icon.babysitting.data.price_range = n/a
#icon.babysitting.data.price_range.maxprice = n/a

### action labels
```

Listing A.2: Label properties for generated user interface, domain ”babysitting”

Data Description Model and Label Properties for Domain "Ticketing"

```
data_description : "concert"
referencedOntologies [
base: "http://xxx.org/ticketingDO/v1#"
tiod: "http://xxx.org/ticketing/v1#"
gr : "http://purl.org/goodrelations/v1#"
xmls: "http://www.w3.org/2001/XMLSchema#"
]
{
group: "concertdata" { swClass="TicketRequest"
swIndividual="ticketrequest"
}
{
group: "concertdetails"{
name:    {  swForIndividual="ticketrequest"
swProperty="gr:name"
swType="gr:name"
}
concertcategory:  {
restrictedTo="musical|classical|rock|jazz|all"
initialValue="all"
type="manyOfMany"
swForIndividual="ticketrequest"
swProperty="tiod:eventcategory"
swType="tiod:eventcategorylist"
}
}
group: "pricerange" {
swClass="gr:UnitPriceSpecification"
swIndividual="hasPriceSpecification"
swForIndividual="ticketrequest"
swProperty="gr:hasPriceSpecification"
}
{
maxprice: {
type="number", unit="EUR"
swForIndividual="hasPriceSpecification",
swProperty="gr:hasMaxCurrencyValue",
swType="xmls:float"
}
}
}/*
group :"data" {}
{
summary: { type="snippet"
initialValue="snippets/datamodel.html"
}
}
*/
}
```

Listing A.3: Data description model for domain "ticketing"

```

### field labels
concert = Concert information
concert.concertdata = Concert information
concert.concertdata.concertdetails = Details for title or genre
concert.concertdata.concertdetails.concertcategory = Concert Category
concert.concertdata.concertdetails.name = Please enter a title
concert.concertdata.pricerange = Price range
concert.concertdata.pricerange.maxprice = Maximum Price

### value labels for restricted fields
values.concert.concertdata.concertdetails.concertcategory = Musical|
    Classical|Rock|Jazz|Other
####values: musical|classical|rock|jazz|all
#values.infos.concert.concertdata.concertdetails.concertcategory = n/a
#values.icons.concert.concertdata.concertdetails.concertcategory = n/a


### field infos
info.concert = Please tell us your preferences
#info.concert.concertdata = n/a
#info.concert.concertdata.concertdetails = n/a
#info.concert.concertdata.concertdetails.concertcategory = n/a
#info.concert.concertdata.concertdetails.name = n/a
info.concert.concertdata.pricerange = Please provide a price range for
    your tickets
#info.concert.concertdata.pricerange.maxprice = n/a
```

```

### field icons
#icon.concert = n/a
#icon.concert.concertdata = n/a
#icon.concert.concertdata.concertdetails = n/a
#icon.concert.concertdata.concertdetails.concertcategory = n/a
#icon.concert.concertdata.concertdetails.name = n/a
#icon.concert.concertdata.pricerange = n/a
#icon.concert.concertdata.pricerange.maxprice = n/a

### action labels
```

Listing A.4: Label properties for domain "ticketing"

Data Description Model and Label Properties for Domain "Gastronomy"

```
data_description : "restaurant"
referencedOntologies [
base: "http://xxx.org/ticketingDO/v1#"
tiod: "http://xxx.org/ticketing/v1#"
gr : "http://purl.org/goodrelations/v1#"
xmls: "http://www.w3.org/2001/XMLSchema#"
]
{
group: "data" { swClass="RestaurantRequest"
swIndividual="restaurantrequest"
/*
???
*/
}
{
group: "restaurantdetails"{
name:     {  swForIndividual="restoranrequest"
swProperty="gr:name"
swType="gr:name"
}


prefered cuisine:{
restrictedTo="asia|italian|fransh|mexican|all"
initialValue="all"
type="manyOfMany"
swForIndividual="restoranrequest"
swProperty="tiod:eventcategory"
swType="tiod:eventcategorylist"
/*
swForIndividual="restoranrequest"
swProperty="pod:preferedcategory"
swType="pod:preferedcategorylist"
*/
}
}
group: "reservation" {
swClass="gr:UnitPriceSpecification"
swIndividual="hasPriceSpecification"
swForIndividual="restoranrequest"
swProperty="gr:hasPriceSpecification"
/*
???
*/

}
{
for: {
type="number", unit=""
initialValue="2"
swForIndividual="hasPriceSpecification",
swProperty="gr:hasMaxCurrencyValue",
swType="xmls:float"
/*
???
*/
}
}
}/*
group :"data" {}
{
summary: { type="snippet"
initialValue="snippets/datamodel.html"
}
}
*/
}
```

Listing A.5: Data description model for domain "gastronomy"

```

### field labels
restaurant = Restaurant
restaurant.data = data
restaurant.data.reservation = Make a reservation
restaurant.data.reservation.for = for how many persons?
restaurant.data.restaurantdetails = Some deatails about the reservation
restaurant.data.restaurantdetails.name = the name of the restaurant ...
restaurant.data.restaurantdetails.prefered = prefered cuisine

### value labels for restricted fields
values.restaurant.data.restaurantdetails.prefered = asian|italian|
    fransh|mexican|all
#values: asia|italian||fransh|mexican|all
#values.infos.restaurant.data.restaurantdetails.prefered = n/a
#values.icons.restaurant.data.restaurantdetails.prefered = n/a


```

```
### field infos
#info.restaurant = n/a
#info.restaurant.data = n/a
#info.restaurant.data.reservation = n/a
#info.restaurant.data.reservation.for = n/a
#info.restaurant.data.restaurantdetails = n/a
#info.restaurant.data.restaurantdetails.name = n/a
#info.restaurant.data.restaurantdetails.prefered = n/a

### field icons
#icon.restaurant = n/a
#icon.restaurant.data = n/a
#icon.restaurant.data.reservation = n/a
#icon.restaurant.data.reservation.for = n/a
#icon.restaurant.data.restaurantdetails = n/a
#icon.restaurant.data.restaurantdetails.name = n/a
#icon.restaurant.data.restaurantdetails.prefered = n/a
```

Listing A.6: Label properties for domain "gastronomy"

Data Description Model and Label Properties for Domain "Parking"

```
data_description : "parking"
referencedOntologies [
base: "http://xxx.org/ticketingDO/v1#"
tiod: "http://xxx.org/ticketing/v1#"
gr : "http://purl.org/goodrelations/v1#"
xmls: "http://www.w3.org/2001/XMLSchema#"
]
{
group: "information" { swClass="ParkingRequest"
swIndividual="parkingrequest"
/*
???
*/
}
{
group: "parkingdetails"{
/* name:    {swForIndividual="parkingrequest"
swProperty="gr:name"
swType="gr:name"
*/
}
prefered:{
restrictedTo="parking|garage|disabled|all"
initialValue="all"
type="manyOfMany"
swForIndividual="parkingrequest"
swProperty="tiod:eventcategory"
swType="tiod:eventcategorylist"
/*
swForIndividual="parkingrequest"
swProperty="pod:preferedcategory"
swType="pod:preferedcategorylist"
*/
}
}
group: "price_range" {
swClass="gr:UnitPriceSpecification"
swIndividual="hasPriceSpecification"
swForIndividual="parkingrequest"
swProperty="gr:hasPriceSpecification"
/*
???
*/

}
{
maxprice: {
type="number", unit="EUR"
initialValue="1.1"
swForIndividual="hasPriceSpecification",
swProperty="gr:hasMaxCurrencyValue",
swType="xmls:float"
/*
???
*/
}
}
}/*
group :"data" {}
{
summary: { type="snippet"
initialValue="snippets/datamodel.html"
}
}
*/
}
```

Listing A.7: Data description model for domain "parking"

```

### field labels
concert = Concert information
concert.concertdata = Concert information
concert.concertdata.concertdetails = Details for title or genre
concert.concertdata.concertdetails.concertcategory = Concert Category
concert.concertdata.concertdetails.name = Please enter a title
concert.concertdata.pricerange = Price range
concert.concertdata.pricerange.maxprice = Maximum Price

### value labels for restricted fields
values.concert.concertdata.concertdetails.concertcategory = Musical|
    Classical|Rock|Jazz|Other
####values: musical|classical|rock|jazz|all
#values.infos.concert.concertdata.concertdetails.concertcategory = n/a
```

```
#values.icons.concert.concertdata.concertdetails.concertcategory = n/a


### field infos
info.concert = Please tell us your preferences
#info.concert.concertdata = n/a
#info.concert.concertdata.concertdetails = n/a
#info.concert.concertdata.concertdetails.concertcategory = n/a
#info.concert.concertdata.concertdetails.name = n/a
info.concert.concertdata.pricerange = Please provide a price range for
    your tickets
#info.concert.concertdata.pricerange.maxprice = n/a

### field icons
#icon.concert = n/a
#icon.concert.concertdata = n/a
#icon.concert.concertdata.concertdetails = n/a
#icon.concert.concertdata.concertdetails.concertcategory = n/a
#icon.concert.concertdata.concertdetails.name = n/a
#icon.concert.concertdata.pricerange = n/a
#icon.concert.concertdata.pricerange.maxprice = n/a
```

Listing A.8: Label properties for domain "parking"

A.2 Catalog of Providers

```

@prefix bns:   <http://business-name-system.org/base/> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix v:     <http://schema.org/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gr:    <http://purl.org/goodrelations/v1#> .

<http://business-name-system.org/base>
bns:description  "Business␣Domain␣System" ;
bns:hasDomain    <http://business-name-system.org/babysitting> , <http
    ://business-name-system.org/parking> , <http://business-name-system.
    org/gastronomy> , <http://business-name-system.org/ticketing> .

<http://business-name-system.org/babysitting>
a              "domain" ;
rdfs:comment   "babysitting␣..." ;
rdfs:label     "babysitting" ;
bns:hasEntity  <http://business-name-system.org/entity/LittleMonkeys> .

<http://business-name-system.org/entity/LittleMonkeys>
rdfs:comment            "Bilingual␣English␣and␣German␣nursery,␣with␣
    garden,␣for␣1-3␣year␣olds." ;
rdfs:label              "Little␣Monkeys" ;
gr:BusinessEntity       [ bns:hasGlobalIdentifier  "60311-FFM-98797" ;
bns:hasPeerID            "205w270gw8p30udi" ;
gr:address                <http://business-name-system.org/entity/LittleMonkeys/address> ;
gr:category               "Nanny␣Services" ;
gr:description            "Bilingual␣English␣and␣German␣nursery,␣with␣garden,␣for␣1-3␣year␣olds." ;
gr:legalName              "Little␣Monkeys␣GmbH" ;
v:email                   "info@frankfurt-little-monkeys.de" ;
v:telephone               "069-5555-555" ;
v:web                     "http://www.frankfurt-little-monkeys.de"
] ;
gr:hasBusinessFunction  "sell" .


<http://business-name-system.org/entity/LittleMonkeys/address>
v:postalAddress  <http://business-name-system.org/entity/LittleMonkeys/postalAddress> .

<http://business-name-system.org/entity/LittleMonkeys/postalAddress>
v:addressLocality  "␣␣Frankfurt␣am␣Main␣" ;
v:postalCode       "␣␣60318␣␣" ;
v:streetAddress    "␣␣Glauburgstr.␣36␣␣" .

<http://business-name-system.org/parking>
a              "domain" ;
rdfs:comment   "parking␣..." ;
rdfs:label     "parking" ;
bns:hasEntity  <http://business-name-system.org/entity/ParkhausMyZeil> .

<http://business-name-system.org/entity/ParkhausMyZeil>
rdfs:comment            "Parking␣Garage␣in␣Frankfurt␣downtown" ;
rdfs:label              "Parkhaus␣MyZeil" ;
gr:BusinessEntity       [ bns:hasGlobalIdentifier  "60432-FFM-00101" ;
bns:hasPeerID            "08m243vf8zzadcxr" ;
gr:address               <http://business-name-system.org/entity/ParkhausMyZeil/address> ;
gr:category              "Garage" ;
gr:description           "Parking␣Garage␣in␣Frankfurt␣downtown" ;
gr:legalName             "Parkhaus␣MyZeil␣-␣PalaisQuartier␣APCOA" ;
v:email                  "info@PalaisQuartierAPCOA.de" ;
v:telephone              "0711␣30570305" ;
v:web                    "http://www.PalaisQuartierAPCOA.de/MyZeil"
] ;
gr:hasBusinessFunction  "sell" .

<http://business-name-system.org/entity/ParkhausMyZeil/address>
v:postalAddress  <http://business-name-system.org/entity/ParkhausMyZeil/postalAddress> .

<http://business-name-system.org/entity/ParkhausMyZeil/postalAddress>
v:addressLocality  "␣␣Frankfurt␣␣" ;
v:postalCode       "␣␣60432␣␣" ;
v:streetAddress    "␣␣Große␣Eschenheimer␣Str.␣10␣␣" .

<http://business-name-system.org/gastronomy>
```

```
a               "domain" ;
rdfs:comment    "gastronomy␣..." ;
rdfs:label      "gastronomy" ;
bns:hasEntity  <http://business-name-system.org/entity/
    RestaurantTrattoriaFantastico> .

<http://business-name-system.org/entity/RestaurantTrattoriaFantastico>
rdfs:comment              "Italian␣Two-Star␣Restaurant␣in␣Frankfurt␣
    downtown" ;
rdfs:label                "Trattoria␣Fantastico" ;
gr:BusinessEntity        [ bns:hasGlobalIdentifier  "60311-FFM-12345" ;
bns:hasPeerID             "s4621ig6hnfxn7b9" ;
gr:address                <http://business-name-system.org/entity/
    RestaurantTrattoriaFantastico/address> ;
gr:category               "Itallian" ;
gr:description            "Italian␣Two-Star␣Restaurant␣in␣Frankfurt␣
    downtown" ;
gr:legalName              "Restaurant␣Trattoria␣Fantastico␣GmbH" ;
v:email                   "info@tratoria-fantastico.de" ;
v:telephone               "069-1570070" ;
v:web                     "http://www.tratoria-fantastico.de"
] ;
gr:hasBusinessFunction  "sell" .

<http://business-name-system.org/entity/RestaurantTrattoriaFantastico/
    address>
v:postalAddress  <http://business-name-system.org/entity/
    RestaurantTrattoriaFantastico/postalAddress> .

<http://business-name-system.org/entity/RestaurantTrattoriaFantastico/
    postalAddress>
v:addressLocality  "␣␣Frankfurt␣␣" ;
v:postalCode       "␣␣65432␣␣" ;
v:streetAddress    "␣␣Fressgass␣␣" .

<http://business-name-system.org/ticketing>
a               "domain" ;
rdfs:comment    "ticketing␣..." ;
rdfs:label      "ticketing" ;
bns:hasEntity  <http://business-name-system.org/entity/AlteOper> , <
    http://business-name-system.org/entity/eventim> .

<http://business-name-system.org/entity/AlteOper>
rdfs:comment              "Alte␣Oper␣..." ;
rdfs:label                "AlteOper" ;
gr:BusinessEntity        [ bns:hasGlobalIdentifier  "99zU67FF42" ;
bns:hasPeerID             "gftp8vwcekfyldi" ;
gr:address                <http://business-name-system.org/entity/
    AlteOper/address> ;
gr:category               "Oper" ;
gr:description            "␣␣...␣␣" ;
gr:legalName              "Alte␣Oper␣Frankfurt" ;
v:email                   "alte@oper.de" ;
v:telephone               "+49␣63␣1234545"
] ;
```

```
gr:hasBusinessFunction  "sell" .

<http://business-name-system.org/entity/AlteOper/address>
v:postalAddress  <http://business-name-system.org/entity/AlteOper/
   postalAddress> .

<http://business-name-system.org/entity/AlteOper/postalAddress>
v:addressLocality  "␣␣Frankfurt␣␣" ;
v:postalCode       "␣␣65432␣␣" ;
v:streetAddress    "␣␣Operplatz␣␣" .

<http://business-name-system.org/entity/eventim>
rdfs:comment            "Europas␣führendes␣Ticketing-␣und␣Live-
   Entertainment-Unternehmen." ;
rdfs:label              "Eventim" ;
gr:BusinessEntity       [ bns:hasGlobalIdentifier  "80333-M-75765" ;
bns:hasPeerID            "w6sge5hfxenrk9" ;
gr:address               <http://business-name-system.org/entity/
   eventim/address> ;
gr:category              "Tickets" ;
gr:description           "CTS␣EVENTIM␣AG␣and␣Co.␣KGaA␣is␣Europas␣fü
   hrendes␣Ticketing-␣und␣Live-Entertainment-Unternehmen." ;
gr:legalName             "CTS␣EVENTIM␣AG␣and␣Co.␣KGaA" ;
v:email                  "info@eventim.de" ;
v:telephone              "01806-570070" ;
v:web                    "http://www.eventim.de"
] ;
gr:hasBusinessFunction  "sell" .

<http://business-name-system.org/entity/eventim/address>
v:postalAddress  <http://business-name-system.org/entity/eventim/
   postalAddress> .

<http://business-name-system.org/entity/eventim/postalAddress>
v:addressLocality  "␣␣Munich␣␣" ;
v:postalCode       "␣␣80333␣␣" ;
v:streetAddress    "␣␣Adenauer␣Str.␣␣" .
```

Listing A.9: Initial catalog of providers

A.3 Complex Product Request

```
{
"@": "@prefix␣:␣.
@prefix␣owl:␣.
@prefix␣rdf:␣.
@prefix␣xml:␣.
@prefix␣xsd:␣.
@prefix␣rdfs:␣.
@base␣.

rdf:type␣owl:Ontology␣.

#################################################################
#␣Datatypes
#################################################################

###␣xmls:float
rdf:type␣rdfs:Datatype␣.

###␣xmls:number
rdf:type␣rdfs:Datatype␣.

###␣xmls:preferred
rdf:type␣rdfs:Datatype␣.

###␣xmls:text
rdf:type␣rdfs:Datatype␣.

#################################################################
#␣Object␣Properties
#################################################################

###␣http://mimesis.solutions//babysitting/individuals#data
:data␣rdf:type␣owl:ObjectProperty␣.

###␣http://purl.org/goodrelations/v1#hasPriceSpecification
rdf:type␣owl:ObjectProperty␣.

#################################################################
#␣Data␣properties
#################################################################

###␣http://bns.farberg.de/demo/base#domain
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/babysitting/v1#age
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/babysitting/v1#availability
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/babysitting/v1#duration
rdf:type␣owl:DatatypeProperty␣.
```

```

###␣http://products.org/babysitting/v1#pickuptime
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/babysitting/v1#preferred
rdf:type␣owl:DatatypeProperty␣.

###␣http://purl.org/goodrelations/v1#hasMaxCurrencyValue
rdf:type␣owl:DatatypeProperty␣.

#################################################################
#␣Classes
#################################################################

###␣BabysittingRequest
rdf:type␣owl:Class␣.

###␣gr:UnitPriceSpecification
rdf:type␣owl:Class␣.

#################################################################
#␣Individuals
#################################################################

###␣http://mimesis.solutions//babysitting/individuals#_i1496335592986
:_i1496335592986␣rdf:type␣owl:NamedIndividual␣;
:data␣:babysittingrequest_i1496335592986␣.

###␣http://mimesis.solutions//babysitting/individuals#
    babysittingrequest_i1496335592986
:babysittingrequest_i1496335592986␣rdf:type␣,
owl:NamedIndividual␣;

\"1-3␣years\"^^␣;
\"16\"^^␣;
\"6\"^^␣;
\">40\"^^␣;
\"babysitting\"^^␣;
\"evening\"^^␣;
\"|babysitter\"^^␣;
:hasPriceSpecification_i1496335592986␣.

###␣http://mimesis.solutions//babysitting/individuals#
    hasPriceSpecification_i1496335592986
:hasPriceSpecification_i1496335592986␣rdf:type␣,
owl:NamedIndividual␣;
\"15\"^^␣.

###␣Generated␣by␣the␣OWL␣API␣(version␣0.0.1-SNAPSHOT)␣http://owlapi.
    sourceforge.net
",
"1": "@prefix␣:␣.
@prefix␣owl:␣.
@prefix␣rdf:␣.
@prefix␣xml:␣.
```

```
@prefix␣xsd:␣.
@prefix␣rdfs:␣.
@base␣.

rdf:type␣owl:Ontology␣.

#################################################################
#␣Datatypes
#################################################################

###␣gr:name
rdf:type␣rdfs:Datatype␣.

###␣tiod:eventcategorylist
rdf:type␣rdfs:Datatype␣.

###␣xmls:float
rdf:type␣rdfs:Datatype␣.

###␣xmls:text
rdf:type␣rdfs:Datatype␣.

#################################################################
#␣Object␣Properties
#################################################################

###␣http://mimesis.solutions//concert/individuals#concertdata
:concertdata␣rdf:type␣owl:ObjectProperty␣.

###␣http://purl.org/goodrelations/v1#hasPriceSpecification
rdf:type␣owl:ObjectProperty␣.

#################################################################
#␣Data␣properties
#################################################################

###␣http://bns.farberg.de/demo/base#domain
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/ticketing/v1#eventcategory
rdf:type␣owl:DatatypeProperty␣.

###␣http://purl.org/goodrelations/v1#hasMaxCurrencyValue
rdf:type␣owl:DatatypeProperty␣.

###␣http://purl.org/goodrelations/v1#name
rdf:type␣owl:DatatypeProperty␣.

#################################################################
#␣Classes
#################################################################

###␣TicketRequest
rdf:type␣owl:Class␣.

```

```
###␣gr:UnitPriceSpecification
rdf:type␣owl:Class␣.

#################################################################
#␣Individuals
#################################################################

###␣http://mimesis.solutions//concert/individuals#_i1496335592977
:_i1496335592977␣rdf:type␣owl:NamedIndividual␣;
:concertdata␣:ticketrequest_i1496335592977␣.

###␣http://mimesis.solutions//concert/individuals#
    hasPriceSpecification_i1496335592977
:hasPriceSpecification_i1496335592977␣rdf:type␣,
owl:NamedIndividual␣;
\"40\"^^␣.

###␣http://mimesis.solutions//concert/individuals#
    ticketrequest_i1496335592977
:ticketrequest_i1496335592977␣rdf:type␣,
owl:NamedIndividual␣;
\"Chicago\"^^␣;
\"ticketing\"^^␣;
\"|musical|all\"^^␣;
:hasPriceSpecification_i1496335592977␣.

###␣Generated␣by␣the␣OWL␣API␣(version␣0.0.1-SNAPSHOT)␣http://owlapi.
    sourceforge.net
",
"2": "@prefix␣:␣.
@prefix␣owl:␣.
@prefix␣rdf:␣.
@prefix␣xml:␣.
@prefix␣xsd:␣.
@prefix␣rdfs:␣.
@base␣.

rdf:type␣owl:Ontology␣.

#################################################################
#␣Datatypes
#################################################################

###␣gr:name
rdf:type␣rdfs:Datatype␣.

###␣tiod:parkingpreferencesList
rdf:type␣rdfs:Datatype␣.

###␣xmls:float
rdf:type␣rdfs:Datatype␣.

#################################################################
#␣Object␣Properties
#################################################################
```

```

###␣http://mimesis.solutions//parking/individuals#information
:information␣rdf:type␣owl:ObjectProperty␣.

###␣http://purl.org/goodrelations/v1#hasPriceSpecification
rdf:type␣owl:ObjectProperty␣.

#################################################################
#␣Data␣properties
#################################################################

###␣http://bns.farberg.de/demo/base#domain
rdf:type␣owl:DatatypeProperty␣.

###␣http://purl.org/goodrelations/v1#hasMaxCurrencyValue
rdf:type␣owl:DatatypeProperty␣.

###␣http://xxx.org/parking/v1#preferred
rdf:type␣owl:DatatypeProperty␣.

#################################################################
#␣Classes
#################################################################

###␣ParkingRequest
rdf:type␣owl:Class␣.

###␣gr:UnitPriceSpecification
rdf:type␣owl:Class␣.

#################################################################
#␣Individuals
#################################################################

###␣http://mimesis.solutions//parking/individuals#_i1496335592977
:_i1496335592977␣rdf:type␣owl:NamedIndividual␣;
:information␣:parkingrequest_i1496335592977␣.

###␣http://mimesis.solutions//parking/individuals#
    hasPriceSpecification_i1496335592977
:hasPriceSpecification_i1496335592977␣rdf:type␣,
owl:NamedIndividual␣;
\"2\"^^␣.

###␣http://mimesis.solutions//parking/individuals#
    parkingrequest_i1496335592977
:parkingrequest_i1496335592977␣rdf:type␣,
owl:NamedIndividual␣;
\"parking\"^^␣;
\"|parking|garage\"^^␣;
:hasPriceSpecification_i1496335592977␣.

###␣Generated␣by␣the␣OWL␣API␣(version␣0.0.1-SNAPSHOT)␣http://owlapi.
    sourceforge.net
",
"3": "@prefix␣:␣.
@prefix␣owl:␣.
@prefix␣rdf:␣.
@prefix␣xml:␣.
@prefix␣xsd:␣.
@prefix␣rdfs:␣.
@base␣.

rdf:type␣owl:Ontology␣.

#################################################################
#␣Datatypes
#################################################################

###␣tiod:foodcategorylist
rdf:type␣rdfs:Datatype␣.

###␣tiod:locationname
rdf:type␣rdfs:Datatype␣.

###␣xmls:number
rdf:type␣rdfs:Datatype␣.

###␣xmls:text
rdf:type␣rdfs:Datatype␣.

#################################################################
#␣Object␣Properties
#################################################################

###␣http://mimesis.solutions//restaurant/individuals#data
:data␣rdf:type␣owl:ObjectProperty␣.

#################################################################
#␣Data␣properties
#################################################################

###␣http://bns.farberg.de/demo/base#domain
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/ticketing/v1#foodcategory
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/ticketing/v1#name
rdf:type␣owl:DatatypeProperty␣.

###␣http://products.org/ticketing/v1#persons
rdf:type␣owl:DatatypeProperty␣.

#################################################################
#␣Classes
#################################################################

###␣RestaurantRequest
rdf:type␣owl:Class␣.
```

```

#################################################################
#␣Individuals
#################################################################

###␣http://mimesis.solutions//restaurant/individuals#_i1496335592980
:_i1496335592980␣rdf:type␣owl:NamedIndividual␣;
:data␣:restaurantrequest_i1496335592980␣.

###␣http://mimesis.solutions//restaurant/individuals#restaurantrequest_i1496335592980
:restaurantrequest_i1496335592980␣rdf:type␣,
owl:NamedIndividual␣;
\"4\"^^␣;
\"Da␣Guido\"^^␣;
\"gastronomy\"^^␣;
\"|italian\"^^␣.

###␣Generated␣by␣the␣OWL␣API␣(version␣0.0.1-SNAPSHOT)␣http://owlapi.sourceforge.net
"
}
```

Listing A.10: Complex product request as the resulting RDF document

Bibliography

[Adn17] Ron Adner. “Ecosystem as structure: an actionable construct for strategy”. In: *Journal of Management* 43.1 (2017), pp. 39–58.

[AF14] Leonidas Anthopolous and Panos Fitsilis. “Exploring Architectural and Organizational Features in Smart Cities”. In: (2014). DOI: 10.1109/ICACT.2014.6778947.

[AG07] Frederik Ahlemann and Heike Gastl. “Process model for an empiracally grounded reference model construction”. In: *Reference modeling for business systems analysis* (2007), pp. 77–97.

[AH00] Eytan Adar and Bernardo A Huberman. “Free riding on Gnutella”. In: *First monday* 5.10 (2000).

[ale] Amazon alexa. *Siri*. URL: https://developer.amazon.com/de-DE/alexa (visited on 03/21/2022).

[Alh21] Fawaz Alharbi. “The Use of Digital Healthcare Platforms During the COVID-19 Pandemic: the Consumer Perspective”. In: *Acta Informatica Medica* 29.1 (2021), p. 51.

[And06] Chris Anderson. *The long tail*. Hyperion, 2006.

[Ant16] James J. Anton. “Adverse Selection”. In: *The Palgrave Encyclopedia of Strategic Management*. Ed. by Mie Augier and J. David Teece. London: Palgrave Macmillan UK, 2016, pp. 1–2.

[Ass08] DBpedia Association. *DBPedia Ontology*. 2008. URL: https://www.dbpedia.org/resources/ontology/ (visited on 03/21/2022).

[Aul+16] Fabian Aulkemeier et al. “A service-oriented e-commerce reference architecture”. In: *Journal of theoretical and applied electronic commerce research* 11.1 (2016), pp. 26–45.

[AW18] Andreas M Antonopoulos and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and Dapps*. O’Reilly Media, 2018.

[Bak97] Yannis Bakos. “Reducing buyer search costs: Implications for electronic marketplaces”. In: *Management science* 43.12 (1997), pp. 1676–1692.

[Bat+13] Cinzia Battistella et al. “Methodology of business ecosystems network analysis: A case study in Telecom Italia Future Centre”. In: *Technological Forecasting and Social Change* 80.6 (2013), pp. 1194–1210.

[Baz16] Open Bazaar. *Open Bazaar*. 2016. URL: https://blockchain-infos.de/openbazaar/ (visited on 03/21/2022).

[BBC15] BBC. *BBC Core Concepts Ontology*. 2015. URL: http://www.bbc.co.uk/ontologies/coreconcepts (visited on 03/21/2022).

[BDK04] Jörg Becker, Patrick Delfmann, and Ralf Knackstedt. "Konstruktion von Referenzmodellierungssprachen Ein Ordnungsrahmen zur Spezifikation von Adaptionsmechanismen für Informationsmodelle". In: *Wirtschaftsinformatik* 46.4 (2004), pp. 251–264.

[BDK07] Jörg Becker, Patrick Delfmann, and Ralf Knackstedt. "Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models". In: *Reference modeling*. Springer, 2007, pp. 27–58.

[BE07] Robert Braun and Werner Esswein. "Classification of reference models". In: *Advances in Data Analysis*. Springer, 2007, pp. 401–408.

[Ben06] Yochai Benkler. *The wealth of networks: How social production transforms markets and freedom*. Yale University Press, 2006.

[Ber15] Berkman Center for Internet and Society at Harvard University. *Project VRM*. 2015. URL: http://blogs.law.harvard.edu/vrm/projects/ (visited on 03/21/2022).

[Ber17] Tim Berners-Lee. *The SOLID Project*. 2017. URL: https://solid.inrupt.com (visited on 03/21/2022).

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. "The semantic web". In: *Scientific american* 284.5 (2001), pp. 34–43.

[Bit] Bittorrent. *Bittorrent*. URL: https://www.bittorrent.com/ (visited on 02/19/2022).

[Bit11] Bitcoin. *Bitcoin*. 2011. URL: https://bitcoin.org/ (visited on 06/01/2021).

[BKF17] Dennis Boldt, Felix Kaminski, and Stefan Fischer. "Decentralized Bootstrapping for WebRTC-based P2P Networks". In: *The Fifth International Conference on Building and Exploring Web Based Environments (WEB2017)*. 2017, pp. 17–23.

[Bro07] Jan vom Brocke. "Design principles for reference modeling: reusing information models by means of aggregation, specialisation, instantiation, and analogy". In: *Reference modeling for business systems analysis*. IGI Global, 2007, pp. 47–76.

[BRS13] Jörg Becker, Michael Rosemann, and Reinhard Schütte. *Referenzmodellierung: State-of-the-Art und Entwicklungsperspektiven*. Springer-Verlag, 2013.

[BRS95] Jörg Becker, Michael Rosemann, and Reinhard Schütte. "Guidelines of modelling (GoM)". In: *Wirtschaftsinformatik* 37.5 (1995), pp. 435–445.

[BS04] Jörg Becker and Reinhard Schütte. *Handelsinformationssysteme*. MI Wirtschaftsbuch, 2004.

[BS07] Jorg Becker and Reinhard Schutte. "A Reference Model for Retail Enterprise". In: *Reference Modeling for Business Systems Analysis*. IGI Global, 2007, pp. 182–205.

[Bun] 1045–1077 Bundesgesetzblatt 2000. *Gesetz zur Verhütung und Bekämpfung von Infektionskrankheiten beim Menschen, Infektionsschutzgesetz – IfSG*.

[Bun20] Bundesministerium für Gesundheit. *Nationale Impfstrategie COVID-19: Strategie zur Einführung und Evaluierung einer Impfung gegen Sars-CoV-2 in Deutschland*. 2020. URL: `http://www.bundesgesundheitsministerium.de/fileadmin/Dateien/3_Downloads/C/Coronavirus/Impfstoff/Nationale_Impfstrategie.pdf` (visited on 03/21/2022).

[CA07] Luis M Camarinha-Matos and Hamideh Afsarmanesh. "A comprehensive modeling framework for collaborative networked organizations". In: *Journal of Intelligent Manufacturing* 18.5 (2007), pp. 529–542.

[CDK05] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[CDN13] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. "Smart cities in Europe". In: *Smart cities*. Routledge, 2013, pp. 185–207.

[CGY19] Michael A Cusumano, Annabelle Gawer, and David B Yoffie. *The business of platforms: Strategy in the age of digital competition, innovation, and power*. HarperCollins New York, NY, 2019.

[Cis13] Cisco. *The Internet of Everything for Cities*. 2013. URL: `http://www.cisco.com/web/about/ac79/docs/ps/motm/IoE-Smart-City_PoV.pdf` (visited on 03/21/2022).

[CKK+03] Paul Clements, Rick Kazman, Mark Klein, et al. *Evaluating software architectures*. Tsinghua University Press Beijing, 2003.

[Coa37] Ronald H Coase. "The nature of the firm". In: *economica* 4.16 (1937), pp. 386–405.

[Coh+04] Michael H Cohen et al. *Voice user interface design*. Addison-Wesley Professional, 2004.

[Coh08] Bram Cohen. *The Bittorrent Protocol Specification*. 2008. URL: `https://www.bittorrent.org/beps/bep_0003.html` (visited on 03/21/2022).

[Coma] European Commission. *A Europe fit for the digital age*. URL: `https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age_en/` (visited on 01/08/2022).

[Comb] European Commission. *Digital Markets Act*. URL: `https://eur-lex.europa.eu/legal-content/en/TXT/?qid=1608116887159&uri=COM%3A2020%3A842%3AFIN` (visited on 01/08/2022).

[Com15] European Commission. *The European single market*. Oct. 2015. URL: `https://ec.europa.eu/growth/single-market_en` (visited on 10/19/2019).

[Con15] W3 Consortium. *Semantic Web*. 2015. URL: `http://www.w3.org/standards/semanticweb/` (visited on 03/21/2022).

[CSS] CSS. *Cascading Style Sheets*. URL: `https://www.w3.org/Style/CSS/Overview.en.html` (visited on 03/21/2022).

[cul21] culture4life. *luca App*. 2021. URL: https://www.luca-app.de (visited on 06/01/2021).

[Deb+20] Pragyan Deb et al. “The effect of containment measures on the COVID-19 pandemic”. In: *IMF Working Paper WP/20/159* (2020).

[Deb04] Frank van Harmelen Deborah L. McGuinness. *OWL Web Ontology Language Overview Schema 1.1*. Feb. 2004. URL: http://www.w3.org/TR/owl-features/ (visited on 03/21/2022).

[Dey01] Anind K Dey. “Understanding and using context”. In: *Personal and ubiquitous computing* 5.1 (2001), pp. 4–7.

[EFL15] Liran Einav, Chiara Farronato, and Jonathan Levin. *Peer-to-Peer Markets*. Tech. rep. National Bureau of Economic Research, 2015.

[EP13] Omar A El Sawy and Francis Pereira. *Business modelling in the dynamic digital space: An ecosystem approach*. Springer, 2013.

[EPV06] Thomas Eisenmann, Geoffrey Parker, and Marshall W Van Alstyne. “Strategies for two-sided markets”. In: *Harvard business review* 84.10 (2006), p. 92.

[EPV11] Thomas Eisenmann, Geoffrey Parker, and Marshall Van Alstyne. “Platform envelopment”. In: *Strategic management journal* 32.12 (2011), pp. 1270–1285.

[Eri14] Gavin Carothers Eric Prud’hommeaux. *RDF 1.1 Turtle*. 2014. URL: http://www.w3.org/TR/turtle/ (visited on 03/21/2022).

[ES16] David S Evans and Richard Schmalensee. *Matchmakers: The new economics of multisided platforms*. Harvard Business Review Press, 2016.

[Eth13] Etherium. *Etherium*. 2013. URL: https://ethereum.org/ (visited on 02/19/2022).

[Eur21] European Commission. *EU Digital COVID Certificate*. 2021. URL: https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_en (visited on 06/01/2021).

[Eym01] Torsten Eymann. “Markets without makers-a framework for decentralized economic coordination in multiagent systems”. In: *Electronic Commerce*. Springer, 2001, pp. 63–74.

[FAS80] William LF Felstiner, Richard L Abel, and Austin Sarat. “The Emergence and Transformation of Disputes: Naming, Blaming, Claiming...” In: *Law and society review* (1980), pp. 631–654.

[Fil+10] Imen Filali et al. “RDF Data Indexing and Retrieval: A survey of Peer-to-Peer based solutions”. PhD thesis. INRIA, 2010.

[FL04] Peter Fettke and Peter Loos. “Referenzmodellierungsforschung”. In: *Wirtschaftsinformatik* 46.5 (2004), pp. 331–340.

[FL07] Peter Fettke and Peter Loos. “Perspectives on reference modeling”. In: *Reference Modeling for Business Systems Analysis*. IGI Global, 2007, pp. 1–21.

[For] Forbes. *Forbes – Global 2000 (2021)*. URL: https://www.forbes.com/lists/global2000/ (visited on 01/08/2022).

[Fou17] Apache Software Foundation. *Apache Jena Components*. 2017. URL: https://jena.apache.org/documentation/ (visited on 03/21/2022).

[Fra04] Ulrich Frank. "E-MEMO: Referenzmodelle zur ökonomischen realisierung leistungsfähiger Infrastrukturen für electronic commerce". In: *Wirtschaftsinformatik* 46.5 (2004), pp. 373–381.

[Fra07] Ulrich Frank. "Evaluation of reference models". In: *Reference modeling for business systems analysis*. IGI Global, 2007, pp. 118–140.

[Gar15] Antonio Garrote. *rdfstore-js*. 2015. URL: https://github.com/antoniogarrote/rdfstore-js (visited on 03/21/2022).

[gem21] gematik. *DEMIS - Deutsches Elektronisches Melde- und Informationssystem für den Infektionsschutz*. 2021. URL: https://confluence-demis.gematik.de/display/DSKB (visited on 06/01/2021).

[GPG14] Claudio Giovanoli, Prasad Pulikal, and S Gatziu Grivas. "E-marketplace for cloud services". In: *Cloud computing* (2014), pp. 76–83.

[Gre15] Paul Grefen. *Beyond E-Business: Towards Networked Structures*. Routledge, 2015, pp. 94–101.

[Gro15] W3C Schema.org Community Group. *Schema.org*. 2015. URL: https://schema.org/docs/about.html (visited on 03/21/2022).

[Hel21] Helmholtz-Zentrum für Infektionsforschung. *SORMAS - Surveillance Outbreak Response Management and Analysis System*. 2021. URL: https://www.sormas-oegd.de/hintergrund (visited on 06/01/2021).

[Hep08] Martin Hepp. "Goodrelations: An ontology for describing products and services offers on the web". In: *Knowledge Engineering: Practice and Patterns*. Springer, 2008, pp. 329–346.

[Hit+16] Michael Hitz et al. "Generic UIs for Requesting Complex Products Within Distributed Market Spaces in the Internet of Everything". In: *Availability, Reliability, and Security in Information Systems, CD-ARES 2016* (2016), pp. 29–44.

[Hit16] Michael Hitz. "mimesis: Ein datenzentrierter Ansatz zur Modellierung von Varianten für Interview-Anwendungen". In: *Multikonferenz Wirtschaftsinformatik* 4 (2016), pp. 1155–1165.

[HKP17] Michael Hitz, Thomas Kessel, and Dennis Pfisterer. "Towards Sharable Application Ontologies for the Automatic Generation of UIs for Dialog based Linked Data Applications." In: *MODELSWARD*. 2017, pp. 65–77.

[HS] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. URL: http://www.w3.org/TR/sparql11-query/ (visited on 03/21/2022).

[HS05] David Hausheer and Burkhard Stiller. “Peermart: The technology for a distributed auction-based market for peer-to-peer services”. In: *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*. Vol. 3. IEEE. 2005, pp. 1583–1587.

[HS09] Volker Hoyer and Katarina Stanoevska-Slabeva. “Towards a reference model for grassroots enterprise mashup environments”. In: (2009).

[HTT] HTTP. *Hypertext Transfer Protocol*. URL: https://www.w3.org/Protocols/ (visited on 03/21/2022).

[HTV06] Erik den Hartigh, Michiel Tol, and Wouter Visscher. “The health measurement of a business ecosystem”. In: *Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting*. 2006, pp. 1–39.

[HW15] Andrei Hagiu and Julian Wright. “Multi-sided platforms”. In: *International Journal of Industrial Organization* 43 (2015), pp. 162–174.

[IL04a] Marco Iansiti and Roy Levien. “Strategy as ecology”. In: *Harvard business review* 82.3 (2004), pp. 68–81.

[IL04b] Marco Iansiti and Roy Levien. *The keystone advantage: what the new dynamics of business ecosystems mean for strategy, innovation, and sustainability*. Harvard Business Press, 2004.

[inc] Apple inc. *Siri*. URL: https://www.apple.com/de/siri/ (visited on 03/21/2022).

[Jav] JavaScript. URL: https://www.javascript.com (visited on 03/21/2022).

[Jena] Apache Jena. *Apache Jena Fuseki*. URL: https://jena.apache.org/documentation/fuseki2/ (visited on 03/21/2022).

[Jenb] Apache Jena. *Apache Jena TDB*. URL: https://jena.apache.org/documentation/tdb/ (visited on 03/21/2022).

[JSO] JSON. *JavaScript Object Notation*. URL: https://www.json.org/json-en.html (visited on 03/21/2022).

[JSS08] Till Janner, Christoph Schroth, and Beat Schmid. “Modelling service systems for collaborative innovation in the enterprise software industry-the st. gallen media reference model applied”. In: *Services Computing, 2008. SCC’08. IEEE International Conference on*. Vol. 2. IEEE. 2008, pp. 145–152.

[Kal00] Christophe Kalt. *Internet relay chat: Client protocol, RFC 2812*. 2000. URL: https://datatracker.ietf.org/doc/html/rfc2812 (visited on 03/21/2022).

[Kao+10] Zoi Kaoudi et al. “Atlas: Storing, updating and querying RDF (S) data on top of DHTs”. In: *Journal of Web Semantics* 8.4 (2010), pp. 271–277.

[Kar+97] David Karger et al. “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web”. In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM. 1997, pp. 654–663.

[Kas21] Kassenärztliche Bundesvereinigung. *Elektronischer Datenaustausch im Disease-Management*. 2021. URL: https://www.kbv.de/html/9537.php (visited on 06/01/2021).

[Kee13] C. Maria Keet. “Open world assumption”. In: *Encyclopedia of Systems Biology* (2013), pp. 1567–1567.

[KHB15] Idehen Kingsley, Martin Hepp, and Aldo Bucchi. *Linked Open Commerce*. 2015. URL: http://linkedopencommerce.com (visited on 03/21/2022).

[KKR01] Ethan Ethan Katsh, M. Ethan Katsh, and Janet Rifkin. *Online dispute resolution: Resolving conflicts in cyberspace*. John Wiley & Sons, Inc., 2001.

[Kle+14] Florian Kleedorfer et al. “The Case for the Web of Needs”. In: *Business Informatics (CBI), 2014 IEEE 16th Conference on*. Vol. 1. IEEE. 2014, pp. 94–101.

[Kle+17] Markus Klems et al. *Desema*. 2017. URL: %7Bhttps://github.com/markusklems/desema%7D (visited on 03/21/2022).

[Kol13] Tobias Kollmann. “E-Business: Grundlagen elektronischer Geschäftsprozesse in der Net Economy. 3”. In: *Auflage. Gabler, Wiesbaden* (2013).

[KPR08] Milan Karow, Daniel Pfeiffer, and Michael Räckers. “Empirical-Based Construction of Reference Models in Public Administrations.” In: *Multikonferenz Wirtschaftsinformatik* (2008), pp. 1613–1624.

[KPS13] N. Komninos, M. Pallot, and H. Schaffers. “Special Issue on Smart Cities and the Future Internet in Europe”. In: *Journal of Knowledge Economy* (2013).

[KS94] Michael L Katz and Carl Shapiro. “Systems competition and network effects”. In: *Journal of economic perspectives* 8.2 (1994), pp. 93–115.

[KZ16] Martin Kenney and John Zysman. “The rise of the platform economy”. In: *Issues in Science and Technology* 32.3 (2016), p. 61.

[Lan21] Landesamt für Gesundheit und Soziales M-V. *Datenverarbeitung für die Schutzimpfung gegen das Corona-Virus*. 2021. URL: https://www.lagus.mv-regierung.de/serviceassistent/download?id=1634270 (visited on 06/01/2021).

[Lim] X/Open Company Limited. *Distributed Transaction Processing: The XA Specification*. Tech. rep. URL: http://pubs.opengroup.org/onlinepubs/009680699/toc.pdf (visited on 03/21/2022).

[LSB03] Seungjoon Lee, Rob Sherwood, and Bobby Bhattacharjee. “Cooperative peer groups in NICE”. In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*. Vol. 2. IEEE. 2003, pp. 1272–1282.

[Luu+16] Loi Luu et al. “Making smart contracts smarter”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 254–269.

[Lux01] Redmer Luxem. *Digital Commerce: electronic commerce mit digitalen Produkten*. Eul, 2001.

[McC14] Tamara McCleary. *Got Influence? What's Social Currency Got To Do With It?* 2014. URL: https://tamaramccleary.com/got-influence-social-currency/ (visited on 03/21/2022).

[Men+12] Andreas Menychtas et al. "A marketplace framework for trading cloud-based services". In: *Economics of Grids, Clouds, Systems, and Services*. Springer, 2012, pp. 76–89.

[Men00] Daniel A Menasce. "Scaling for e-business". In: *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*. IEEE. 2000, pp. 511–513.

[Mer02] Michael Merz. *E-commerce und e-business: marktmodelle, anwendungen und technologien*. dpunkt-Verlag, 2002.

[Mie+13] Richard Mietz et al. "A P2P Semantic Query Framework for the Internet of Things". In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 36.2 (2013), pp. 73–79.

[Mil+02] Dejan Milojicic et al. *Peer-to-peer computing*. 2002.

[MJ16] Alex Moazed and Nicholas L Johnson. *Modern monopolies: What it takes to dominate the 21st century economy*. St. Martin's Press, 2016.

[MMM+04] Frank Manola, Eric Miller, Brian McBride, et al. "RDF primer". In: *W3C recommendation* 10.1-107 (2004), p. 6. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (visited on 03/21/2022).

[Moo05] Daniel L Moody. "Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions". In: *Data & Knowledge Engineering* 55.3 (2005), pp. 243–276.

[Moo93] J.F. Moore. "Predators and Prey: a New Ecology of Competition". In: *Harvard Business Review* 71.3 (1993), pp. 75–86.

[MPK06] Akiyoshi Matono, Said Mirza Pahlevi, and Isao Kojima. "RDFCube: A P2P-based three-dimensional index for structural joins on distributed triple stores". In: *Databases, Information Systems, and Peer-to-Peer Computing*. Springer, 2006, pp. 323–330.

[MS07] Peter Mahlmann and Christian Schindelhauer. *Peer-to-peer-netzwerke: Algorithmen und Methoden*. Springer-Verlag, 2007.

[MZ99] Vojislav B Misic and J Leon Zhao. "Reference models for electronic commerce". In: *Proceedings of the 9th Hong Kong Computer Society Database Conference-Database and Electronic Commerce*. Citeseer. 1999, pp. 109–209.

[Nap01] Napster. *Napster*. 2001. URL: https://ch.napster.com/ (visited on 02/19/2022).

[Neu+20] Karsten Neumann et al. *Future of Health: The rise of Healthcare Platforms*. Sept. 2020. URL: https://www.rolandberger.com/en/Insights/Publications/Future-of-Health-The-rise-of-healthcare-platforms.html (visited on 06/01/2021).

[Nie+15] Sven Niemand et al. "On improving the maintainability of compliance rules for business processes". In: *International Conference on Business Information Systems*. Springer. 2015, pp. 178–190.

[Non94] Martin Nonnenmacher. *Informationsmodellierung unter Nutzung von Referenzmodellen – Die Nutzung von Referenzmodellen zur Implementierung industriebetrieblicher Informationssysteme*. Peter Lang GmbH, Europäischer Verlag der Wissenschaften, 1994.

[Obj15] Inc. Object Management Group. "UML2.5 OMG Unified Modeling Language". In: (2015). URL: http://www.omg.org/spec/UML/2.5/ (visited on 03/21/2022).

[OMG15] OMG. *Business Process Model and Notation Version 2.0*. 2015. URL: http://www.omg.org/spec/BPMN/2.0/ (visited on 06/01/2021).

[Ope16] OpenMarket. *OpenMarket*. 2016. URL: https://github.com/openmarket/openmarket/ (visited on 03/21/2022).

[Ost+11] Alexander Osterwalder et al. "Business Model Generation: A handbook for visionaries, game changers and challengers". In: *African journal of business management* 5.7 (2011), pp. 22–30.

[Ost15] Elinor Ostrom. *Governing the commons*. Cambridge university press, 2015.

[Pap16] BitMarkets White Paper. *BitMarkets*. 2016. URL: https://voluntary.net/bitmarkets/ (visited on 03/21/2022).

[Pee] PeerJS. *PeerJS*. URL: https://peerjs.com (visited on 03/21/2022).

[PRR16] Dennis Pfisterer, Mirjana Radonjic-Simic, and Julian Reichwald. "Business Model Design and Architecture for the Internet of Everything". In: *Journal of Sensor and Actuator Networks* 5.2 (2016), p. 7.

[PRW03] Arnold Picot, Ralf Reichwald und Rolf T Wigand. *Die grenzenlose Unternehmung: Information, Organisation und Management*. Springer-Verlag, 2003.

[PV05] Geoffrey G Parker and Marshall W Van Alstyne. "Two-sided network effects: A theory of information product design". In: *Management science* 51.10 (2005), pp. 1494–1504.

[PVC16] Geoffrey G Parker, Marshall W Van Alstyne, and Sangeet Paul Choudary. *Platform Revolution: How Networked Markets Are Transforming the Economyand How to Make Them Work for You*. WW Norton & Company, 2016.

[Rad+21] Mirjana Radonjic-Simic et al. "Decentralized Open Platform for Vaccination—A German Example: COVID-19-Vacc". In: *Journal of Open Innovation: Technology, Market, and Complexity* 7.3 (2021). ISSN: 2199-8531. DOI: 10.3390/joitmc7030186. URL: https://www.mdpi.com/2199-8531/7/3/186.

[RD01] Antony Rowstron and Peter Druschel. “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems”. In: *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2001, pp. 329–350.

[Rob21] Robert Koch-Institut. *Corona-Warn-App Open-Source-Projekt*. 2021. URL: https://www.coronawarn.app (visited on 06/01/2021).

[Ron+15] Ke Rong et al. “Understanding business ecosystem using a 6C framework in Internet-of-Things-based sectors”. In: *International Journal of Production Economics* 159 (2015), pp. 41–55.

[Ros03] Michael Rosemann. “Application reference models and building blocks for management and control”. In: *Handbook on Enterprise Architecture*. Springer, 2003, pp. 595–615.

[RP15] Mirjana Radonjic-Simic and Dennis Pfisterer. “Requirements for Business Model Design in Context-Centric e-Commerce Environments”. In: *Ausgewählte Aspekte der angewandten Betriebswirtschaftslehre und Wirtschaftsinformatik*. Baumgart, J./Nagler, G.(Hg.) clfmedia Verlag, 2015, 2015, pp. 467–485.

[RP19a] Mirjana Radonjic-Simic and Dennis Pfisterer. “A Decentralized Business Ecosystem for Complex Products”. In: *Digital Business: Business Algorithms, Cloud Computing and Data Engineering*. Springer International Publishing, 2018, 2019.

[RP19b] Mirjana Radonjic-Simic and Dennis Pfisterer. “Beyond Platform Economy: A Comprehensive Model for Decentralized and Self-Organizing Markets on Internet-Scale”. In: *Computers* 8.4 (2019), p. 90.

[RP19c] Mirjana Radonjic-Simic and Dennis Pfisterer. “Reference Model and Architecture for the Post-Platform Economy”. In: *Economics of Digital Transformation*. University of Rijeka, Faculty of Economics and Business, 2019, p. 465.

[RPR20] Mirjana Radonjic-Simic, Dennis Pfisterer, and Petko Rutesic. “Arising Internet of Everything: Business Modeling and Architecture for Smart Cities in Recent Developments in Engineering Research Vol. 8 Chapter: 7”. In: Book Publisher International (a part of SCIENCEDOMAIN International), 2020.

[RR17] Laure Claire Reillier and Benoit Reillier. *Platform Strategy: How to Unlock the Power of Communities and Networks to Grow Your Business*. Routledge, 2017.

[RRP21] Petko Rutesic, Mirjana Radonjic-Simic, and Dennis Pfisterer. “An Enhanced Meta-model to Generate Web Forms for Ontology Population”. In: *Iberoamerican Knowledge Graphs and Semantic Web Conference*. Springer. 2021, pp. 109–124.

[RT03] Jean-Charles Rochet and Jean Tirole. “Platform competition in two-sided markets”. In: *Journal of the european economic association* 1.4 (2003), pp. 990–1029.

[RW11] Nick Rozanski and Eóin Woods. *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley, 2011, pp. 12–29.

[RWP17] Mirjana Radonjic-Simic, Frank Wolff, and Dennis Pfisterer. “Analyzing a Business Ecosystem for Complex Consumer Services”. In: *ICServ2017 - The 5th International Conference on Serviceology: Lecture Notes in Computer Science*. Springer International Publishing, 2017, 2017.

[Sch01] Rüdiger Schollmeier. “A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications”. In: *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*. IEEE. 2001, pp. 101–102.

[Sch03] Markus Schmees. “Distributed digital commerce”. In: *Proceedings of the 5th international conference on Electronic commerce*. ACM. 2003, pp. 131–137.

[Sch13a] Bernhard Schlagheck. *Objektorientierte Referenzmodelle für das Prozess-und Projektcontrolling: Grundlagen—Konstruktion—Anwendungsmöglichkeiten*. Springer-Verlag, 2013.

[Sch13b] Reinhard Schütte. *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations-und anpassungsorientierter Modelle*. Vol. 233. Springer-Verlag, 2013.

[Sch97] Beat Schmid. *The concept of media*. Tech. rep. MCM-institute, University of St. Gallen, 1997.

[Sch98] Reinhard Schütte. “Grundsätze ordnungsmäßiger Referenzmodellierung”. In: *Konstruktion konfigurations-und anpassungsorientierter Modelle. Wiesbaden* (1998).

[Sea13] Doc Searls. *The intention economy: when customers take charge*. Harvard Business Press, 2013.

[Seh21] Sehati. *UniPlat*. 2021. URL: `https://www.sehati.com.sa/uniplat-1` (visited on 06/01/2021).

[Ser+08] Constantin Serban et al. “The concept of decentralized and secure electronic marketplace”. In: *Electronic Commerce Research* 8.1-2 (2008), pp. 79–101.

[SGG01] Stefan Saroiu, P Krishna Gummadi, and Steven D Gribble. “Measurement study of peer-to-peer file sharing systems”. In: *Multimedia Computing and Networking 2002*. Vol. 4673. International Society for Optics and Photonics. 2001, pp. 156–171.

[SJ13] August-Wilhelm Scheer and Wolfram Jost. *ARIS in der Praxis: Gestaltung, Implementierung und Optimierung von Geschäftsprozessen*. Springer-Verlag, 2013.

[SL98] Beat F Schmid and Markus A Lindemann. “Elements of a reference model for electronic markets”. In: *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*. Vol. 4. IEEE. 1998, pp. 193–201.

[SRC84] Jerome H Saltzer, David P Reed, and David D Clark. “End-to-end arguments in system design”. In: *ACM Transactions on Computer Systems (TOCS)* 2.4 (1984), pp. 277–288.

[SS06] Steffen Staab and Heiner Stuckenschmidt. *Semantic Web and Peer-to-peer: decentralized management and exchange of knowledge and information*. Springer, 2006.

[SS08] Beat Schmid and Christoph Schroth. *Organizing as programming: A reference model for cross-organizational collaboration*. 2008.

[SS98] Dorian Selz and Petra Schubert. “Web assessment-a model for the evaluation and the assessment of successful electronic commerce applications”. In: *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*. Vol. 4. IEEE. 1998, pp. 222–231.

[Sta03] Katarina Stanoevska-Slabeva. “Towards a reference model for m-commerce applications”. In: *ECIS 2002 Proceedings* (2003), p. 159.

[Sto+01] Ion Stoica et al. “Chord: A scalable peer-to-peer lookup service for internet applications”. In: *ACM SIGCOMM Computer Communication Review* 31.4 (2001), pp. 149–160.

[Sup05] Association for Supply Chain Management (ASCM). *SCOR-Model*. 2005. URL: `https://www.ascm.org/corporate-transformation/` (visited on 06/01/2021).

[SW05] Ralf Steinmetz and Klaus Wehrle. “2. What Is This “Peer-to-Peer” About?” In: *Peer-to-peer systems and applications*. Springer, 2005, pp. 9–16.

[Sza97] Nick Szabo. “Formalizing and securing relationships on public networks”. In: *First Monday* 2.9 (1997).

[TA14] Llewellyn DW Thomas and Erkko Autio. “The fifth facet: The ecosystem as an organizational field”. In: *Academy of Management Proceedings*. Vol. 2014. 1. Academy of Management. 2014, p. 10306.

[Tee16] David J. Teece. “Business Ecosystem”. In: *The Palgrave Encyclopedia of Strategic Management*. Ed. by Mie Augier and J. David Teece. Palgrave Macmillan UK, 2016.

[Tho06] Oliver Thomas. *Das Referenzmodellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation*. 2006. URL: `https://publikationen.sulb.uni-saarland.de/bitstream/20.500.11880/23828/1/IWi-Heft_187.pdf` (visited on 03/21/2022).

[Tia+08] CH Tian et al. “BEAM: A framework for business ecosystem analysis and modeling”. In: *IBM Systems Journal* 47.1 (2008), pp. 101–114.

[Tiw13] Amrit Tiwana. *Platform ecosystems: Aligning architecture, governance, and strategy*. Newnes, 2013.

[Var08] Stephen L Vargo. “Customer Integration and Value Creation Paradigmatic Traps and Perspectives”. In: *Journal of Service Research* 11.2 (2008), pp. 211–215.

[VLO09] Quang Hieu Vu, Mihai Lupu, and Beng Chin Ooi. *Peer-to-peer computing: Principles and applications*. Springer Science & Business Media, 2009.

[Vom03] Jan Vom Brocke. *Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen*. Vol. 4. Jan vom Brocke, 2003.

[W3C14] W3C. *RDF 1.1 Turtle*. 2014. URL: http://www.w3.org/TR/turtle/ (visited on 03/21/2022).

[Web] WebRTC. *Real-time communication for the Web*. URL: https://webrtc.org (visited on 03/21/2022).

[Wio+19] Frédérique Wiot et al. “Challenges facing vaccinators in the 21st century: results from a focus group qualitative study”. In: *Human vaccines & immunotherapeutics* 15.12 (2019), pp. 2806–2815.

[WLG20] Matthias Walter, Matthias Lohse, and Sabrina Guzman. *Platform Innovation Kit 3.0*. 2020. URL: http://www.platforminnovationkit.com/ (visited on 03/21/2022).

[WLR14] Mika Westerlund, Seppo Leminen, and Mervi Rajahonka. “Designing business models for the internet of things”. In: *Technology Innovation Management Review* 4.7 (2014), p. 5.

[Wol18] Ben Wolford. *What is GDPR, the EU’s new data protection law?* 2018. URL: https://gdpr.eu/what-is-gdpr (visited on 06/01/2021).

[Wor20] International Telecommunication Union World Health Organization. *Digital Health Platform: Building a Digital Information Infrastructure (Infostructure) for Health*. World Health Organization, 2020.

[WSG21] Melanie Windrich, Andreas Speck, and Nils Gruschka. “Representing Data Protection Aspects in Process Models by Coloring”. In: *Annual Privacy Forum*. Springer. 2021, pp. 143–155.

[XL04] Li Xiong and Ling Liu. “Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities”. In: *IEEE transactions on Knowledge and Data Engineering* 16.7 (2004), pp. 843–857.

[XLW10] Yang Xia, Qi Li, and Lei Wang. “Research on Decentralized E-commerce Architecture in P2P Environment”. In: *Electrical and Control Engineering (ICECE), 2010 International Conference on*. IEEE. 2010, pp. 2929–2932.

[Zha+04] Ben Y Zhao et al. “Tapestry: A resilient global-scale overlay for service deployment”. In: *IEEE Journal on selected areas in communications* 22.1 (2004), pp. 41–53.

Own Publications

[Hit+16] Michael Hitz et al. “Generic UIs for Requesting Complex Products Within Distributed Market Spaces in the Internet of Everything”. In: *Availability, Reliability, and Security in Information Systems, CD-ARES 2016* (2016), pp. 29–44.

[PRR16] Dennis Pfisterer, Mirjana Radonjic-Simic, and Julian Reichwald. “Business Model Design and Architecture for the Internet of Everything”. In: *Journal of Sensor and Actuator Networks* 5.2 (2016), p. 7.

[Rad+21] Mirjana Radonjic-Simic et al. “Decentralized Open Platform for Vaccination—A German Example: COVID-19-Vacc”. In: *Journal of Open Innovation: Technology, Market, and Complexity* 7.3 (2021). ISSN: 2199-8531. DOI: `10.3390/joitmc7030186`. URL: `https://www.mdpi.com/2199-8531/7/3/186`.

[RP15] Mirjana Radonjic-Simic and Dennis Pfisterer. “Requirements for Business Model Design in Context-Centric e-Commerce Environments”. In: *Ausgewählte Aspekte der angewandten Betriebswirtschaftslehre und Wirtschaftsinformatik*. Baumgart, J./Nagler, G.(Hg.) clfmedia Verlag, 2015, 2015, pp. 467–485.

[RP19a] Mirjana Radonjic-Simic and Dennis Pfisterer. “A Decentralized Business Ecosystem for Complex Products”. In: *Digital Business: Business Algorithms, Cloud Computing and Data Engineering*. Springer International Publishing, 2018, 2019.

[RP19b] Mirjana Radonjic-Simic and Dennis Pfisterer. “Beyond Platform Economy: A Comprehensive Model for Decentralized and Self-Organizing Markets on Internet-Scale”. In: *Computers* 8.4 (2019), p. 90.

[RP19c] Mirjana Radonjic-Simic and Dennis Pfisterer. “Reference Model and Architecture for the Post-Platform Economy”. In: *Economics of Digital Transformation*. University of Rijeka, Faculty of Economics and Business, 2019, p. 465.

[RPR20] Mirjana Radonjic-Simic, Dennis Pfisterer, and Petko Rutesic. “Arising Internet of Everything: Business Modeling and Architecture for Smart Cities in Recent Developments in Engineering Research Vol. 8 Chapter: 7”. In: Book Publisher International (a part of SCIENCEDOMAIN International), 2020.

[RRP21] Petko Rutesic, Mirjana Radonjic-Simic, and Dennis Pfisterer. “An Enhanced Meta-model to Generate Web Forms for Ontology Population”. In: *Iberoamerican Knowledge Graphs and Semantic Web Conference*. Springer. 2021, pp. 109–124.

[RWP17] Mirjana Radonjic-Simic, Frank Wolff, and Dennis Pfisterer. “Analyzing a Business Ecosystem for Complex Consumer Services”. In: *ICServ2017 - The 5th International Conference on Serviceology: Lecture Notes in Computer Science*. Springer International Publishing, 2017, 2017.