Journal of
Sensor and Actuator Networks

MDPI

Article

Business Model Design and Architecture for the Internet of Everything

Dennis Pfisterer [1,*], Mirjana Radonjic-Simic [2] and Julian Reichwald [2]

[1] Institute of Telematics, University of Lübeck, Lübeck 23562, Germany
[2] Baden-Wuerttemberg Cooperative State University, Mannheim 68163, Germany; mirjana.radonjic-simic@dhbw-mannheim.de (M.R.-S.); julian.reichwald@dhbw-mannheim.de (J.R.)
* Correspondence: pfisterer@itm.uni-luebeck.de; Tel.: +49-621-4105-1373

Academic Editor: Ioannis Chatzigiannakis
Received: 8 January 2016; Accepted: 25 March 2016; Published: 2 April 2016

Abstract: Smart devices and cyber-physical systems, which are interconnected to IT systems and services, form the basis for the arising Internet of Everything, opening up new economic opportunities for its participants and users beyond its technological aspects and challenges. While today's e-business scenarios are mostly dominated by a few centralized online platforms, future business models, which will be feasible for the Internet of Everything, need to address special requirements. Such business models, e.g., leveraging the possibilities of smart cities, need to cope with arbitrary combinations of products and services orchestrated into complex products in a highly distributed and dynamic environment. Furthermore, these arbitrary combinations are influenced by real-time context information derived from sensor networks or IT systems, as well as the users' requirements and preferences. The complexity of finding the optimal product/service combination overstrains users and leads to decisions according to the principle of adverse selection (*i.e.*, choosing good enough instead of optimal). Such e-business models require an appropriate underlying value generation architecture that supports users in this process. In this paper, we develop a business model that addresses these problems. In addition, we present the Distributed Market Spaces (DMS) software-system architecture as a possible implementation, which enables the aforementioned decentralized and context-centric e-business scenario and leverages the commercial possibilities of smart cities.

Keywords: Internet of Everything; smart city; e-business; business models; Industry 4.0

1. Introduction

The Internet has evolved into a global marketplace where information about literally every existing product or service can be found. Since more and more kinds of devices, services, companies, institutions and ultimately people are interconnected to form an *Internet of Everything* [1], the availability of data, products and services will increase significantly by several orders of magnitude. This will give rise to novel types of services and products that have not been possible before. These products and services, as well as complex combinations of them can be tailored to fit the individual needs and contextual requirements of the consumer.

At present, the amount of data, as well as the sum of available products and services in conjunction with possible customizations already overstrain users in search for their optimal choice. This leads to a decision making process according to the principle of adverse selection [2], *i.e.*, choosing good enough instead of optimal. This fact has led to the creation of aggregation and comparison services that support users and potential buyers in making an informed decision. Other offerings extend these services by providing electronic support in the whole transaction process (from supply-demand matching, to payment and settlement, logistics and review). In the last decade, the success of these e-marketplaces has led to a *de facto* centralization of the previously decentralized offerings on the Internet.

As a consequence, these platforms are now in a position where consumers and producers are virtually forced to make use of them and, hence, must accept the rules of these platforms; or they effectively cease to exist in the digital market place. For most consumers (buyers) and providers (sellers), this model works well, as long as they trade individual products and/or certain combinations of them. However, these e-marketplaces are limited in their ability to support users in the case of complex products (*i.e.*, arbitrary combinations of individual products and/or services), which need to fulfill particular conditions (e.g., in relation to a user's context).

As a result, they are not suited as the underlying architecture to support business models in the Internet of Everything where buyer and seller may interact directly with each other without the need for an intermediary. Furthermore, buyer and seller roles are no longer strictly separated, but everybody or everything connected to the Internet can act as a buyer, or a seller, or both at any time. This results in a highly dynamic and complex peer-to-peer structure (a concept called the "peer economy" [3]). This includes, amongst many others, use cases from smart city environments where, e.g., house owners offer short time parking on their property while they are away or tourists are guided through the city based on their individual preferences and possibilities combined with the city's actual contextual state.

In this paper, we present our Distributed Market Spaces (DMS), a software-system architecture that enables distributed, context-centric business models for future e-business scenarios in the peer economy; an e-business environment that supports users (*i.e.*, buyers and sellers):

- in lowering transaction and coordination costs,
- in making commercial transactions of complex products and
- that facilitates the market exchange by alleviating the effects of the growing power of mega-platforms.

The DMS is context-centric (by providing best matching offers for a user's context), highly scalable, strictly decentralized and generic (hence, not being limited to a certain business domain).

This paper is organized as follows. First, Section 2 stipulates the motivation for our work and presents the technological and economical backgrounds. Next, Section 3 discusses the overall objectives, as well as the main functional requirements for a software-system architecture supporting business models in a technology-driven, distributed peer economy. Thereafter, in Section 4, we compare these requirements with current related work. Subsequently, Section 5 discusses the shortcomings of existing approaches and introduces our approach for an underlying architecture, which addresses the requirements and overcomes the shortcomings of existing projects and architectures. Section 6 concludes the paper with a summary and outlook.

2. Motivation and Background

In a smart city, interconnected sensors and actuators are the underlying technology to enable a variety of new use cases and business models [4]. When looking at some of these use cases in detail, especially those incorporating complex services and products, it needs to be understood that most of such scenarios cross the boundaries of a single problem domain and get influenced by many variables and constraints. In this section, we will first look at smart cities (Section 2.1) and business modeling (Section 2.2). Next, we will present an exemplary use case and point out its difficulties in a smart city e-business scenario (Section 2.3). This use case will also serve as the starting point from which to derive the main goals and requirements in the following sections.

2.1. Smart Cities

The promise of smart city projects is mainly to facilitate everyday city life by providing useful services that can be consumed by its inhabitants. Services themselves derive information from data usually gathered from either IT systems or sensors. Therefore, many smart city projects, according to [5], focus on the role of information and communication technology (ICT) as the foundation that allows the creation of a smart city. However, as (smart) cities are "complex social systems and many of them experience difficult issues on the social, economic and environmental domain" [6], they cannot be reduced to their underlying ICT infrastructure. Rather, it is imperative to incorporate environmental and economic aspects of cities, as well.

As per [5], 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 a high quality of life, with a wise management of natural resources, through participatory government" [5]. This definition goes beyond the ICT-centric definitions and covers various aspects of economic and environmental domains found in the literature. However, the definition is vague in terms of *fueling* a high quality of life, economic growth, *etc*. Anthopolous *et al.* [7] follow another approach and define a smart city as "an ICT-based infrastructure and services environment that enhance a city's intelligence, quality of life and other attributes (*i.e.*, environment, entrepreneurship, education, culture, transportation, *etc*.)." This definition covers all economic and environmental aspects found in cities, which can be enhanced by an underlying ICT infrastructure.

In this article, a smart city is regarded as *an ICT-based infrastructure and services environment that enhances a city's intelligence, quality of life and other attributes and adds value to its participants and stakeholders in a context-centric manner*. Note that in this definition, we do not focus on *citizens*, but *participants and stakeholders*, giving citizens the choice of participating in a smart environment. Furthermore, this also includes visitors, tourists, *etc.*, as well as businesses offering higher-level economic services. Such higher-level services can be arbitrarily combined, hence synergistically adding value to the smart city. This is in line with Cisco's definition of the *Internet of Everything* (IoE; *cf.* [1]) that "brings together people, process, data, and things to make networked connections more relevant and valuable".

2.2. e-Business Modeling

As a consequence, the aforementioned definition and the resulting possibilities for participants and stakeholders, smart cities are driven by socio-economic aspects and cannot be reduced to their technological aspect only. A smart city is a highly complex and dynamic ecosystem that allows for new classes of complex *ad hoc* products and services backed and enabled by ICT infrastructure. Therefore, the Internet of Everything (which includes Smart Cities) needs to be looked at not only from a technological perspective, but also from a business perspective. Hence, what is needed is an *e-business model* that integrates the two concurrent forces that reinforce each other: *a technological push force*, representing a technology-driven, novel type of networked business environment, and *a pull force for requirements*, representing the overall objective of a smart city to add value for its participants and stakeholders. Figure 1 shows a conceptual structure of an e-business model (see the description below) and its exposure to technology and forces of the requirements.

The term electronic business (e-business) refers to a business concept that is performed by one or many organizations with the essential use of information technology. According to [8], "e-business is conducting core business activities in a way that is enabled by the integrated use of information technology for processing and communication of information". Thus, an "e-business scenario is considered as a setting in which one or more parties (*i.e.*, peers) engage in e-business to achieve a particular business goal".

The term "business model" refers to a concept that offers a systemic perspective on how to "do business" and focuses on value creation, as well as on value capture [9]. According to [9], a business model depicts the content, structure and governance of transactions designed so as to create value through the exploitation of business opportunities. Emphasizing the system level, business model concepts describe the rationale of a "business system" that lies behind the actual processes [10]. It is centered on activities and focused on value (economic, social, cultural, environmental or other forms of value). According to the definition proposed by [11], a business model encompasses three interconnected and mutually reinforcing building blocks, as can be seen on Figure 1:

- customer value proposition,
- value generation architecture and
- revenue model.

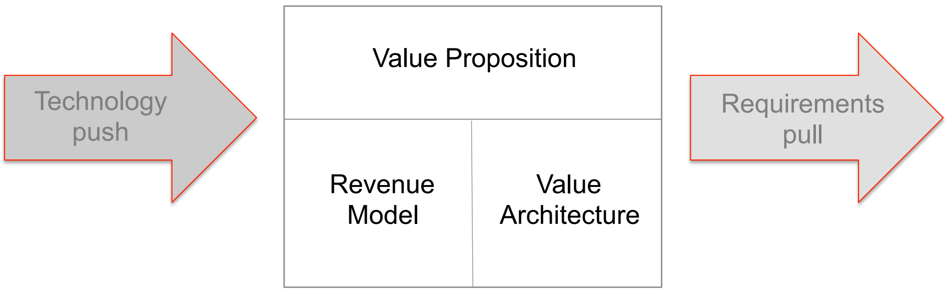


Figure 1. Conceptual structure of a business model.

The customer value proposition describes the bundle of products and/or services that create value for a specific customer segment. The value generation architecture focuses on cross-boundary activities and resources needed for value creation; the revenue model represents the value that an organization generates from each customer segment [12].

There are various approaches in order to formalize business models, such as the business model canvas by Osterwalder *et al.* [10], the business model navigator [13], the value creation framework [14] and the more comprehensive BOAT framework by [8], which we use in this work; the acronym BOAT stands for business (B), organization (O), architecture (A) and technology (T). Unlike the aforementioned approaches, BOAT introduces a method for designing and analyzing e-business models based on a clear and structured separation of concerns by distinguishing between different dimensional aspects: the B and O aspects represent the business-oriented dimension, and the A and T aspects cover the technology-driven dimension needed to support the business dimension of a specific e-business scenario.

In particular, the business aspect describes the nature of the e-business scenario; it deals with the "why" an e-business scenario exists (or should exist) and what should be reached by the collaboration in this specific scenario (*i.e.*, definition of business goals). The organization aspect, on the other hand, deals with the "how" from an organizational perspective, describing how organizations (*i.e.*, involved parties) are structured and connected to achieve the defined business goals. The architecture aspect also deals with "how", but from a conceptual perspective. It provides the high-level structure of the software-system required to support the organizational dimension, thus the defined business goals. Finally, the technological aspect provides the technological details (*i.e.*, software, communication protocols, *etc.*) needed for the realization of a specific e-business scenario.

In this work, we use the BOAT framework to analyze a distributed, context-centric e-business scenario. In order to identify the main business goals (*i.e.*, overall objectives), we first look at the business dimension. Afterwards, we operationalize these objectives into requirements (*i.e.*, business functionalities), which an e-business model needs to fulfill on the operational level. In relation to Figure 1, this represents the force for requirements (*i.e.*, requirements pull), which is discussed in Section 3.

Having analyzed the business and organizational aspects, in Section 5, we then focus on the architecture aspect of the e-business model. Thereby, we create a blueprint of the DMS, a conceptual structure of an underlying value architecture required to make operational aspects work. To this end, we follow the definition of an architecture "*as a set of fundamental concepts of the system in its environment, embodied in its elements, relationships and the principles of its design.*" [15].

2.3. Use Case Description

Consider the use case of a tourist who wants to do certain activities, e.g., visiting the city's theater or a mountaintop close to the city using the mountain railway. Visiting the theater includes a reservation at a nice Italian restaurant and searching for parking close to it, while using the mountain railway is only attractive if the weather is good and the queue in front of it is short. From a commercial viewpoint, a buyer asking for a *complex product*, which incorporates the aforementioned combination, can be satisfied by sellers offering single parts of the complex product. We define a complex product as *any arbitrary combination of products and/or services that add a synergistic value for a buyer based on the user's context*.

Context is understood as a wider range of information to provide a richer description of user's needs and requirements. As such, it can encompass much more information than related to the requested content (*i.e.*, the product and/or service characteristics, composition, *etc.*), decision criteria (*i.e.*, different constraints, like price, configurations, payment and delivery modalities) and include a wider range of relevant information. Among others, these can be user's environmental context information: time, place, temperature or personal preferences (e.g., regional or fair-trade products, specific carbon footprint), reviews and recommendations by trusted social contacts and online communities. Generally speaking, the necessary information is aggregated and derived from data delivered by interconnected IT systems, as well as from sensors and actuator networks, *i.e.*, from the Internet of Things.

While the aforementioned use case is already feasible today, it is representative for a class of problems that tourists and inhabitants of a city face everyday (see also [4] for more examples). Such problems can get arbitrarily complex fairly quickly if more auxiliary conditions, constraints or products/services are added. In order to make informed decisions, users must find their way through a high 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 user involvement lead to the aforementioned phenomenon of adverse selections.

3. Overall Objectives

The scenario presented in the previous section represents a class of e-business use cases that are context-centric, highly distributed and require a large number of fragmented buyers and sellers to collaborate. This demands specific objectives to be fulfilled, thus posing a set of requirements on the conceptual structure of the e-business model. In this section, we analyze such kinds of objectives and discuss the consequential requirements' pull (*cf.* Figure 1). We first look at the main business aspects (Section 3.1) and consider these as the starting point for defining the main business requirements that need to be supported by an underlying value architecture of an e-business model (Section 3.2).

3.1. Business Goals

Considering the business dimension of an e-business scenario, which describes why a specific e-business scenario should exist and what should be achieved by the collaboration in a scenario [8], we formulate what can be seen as *main business goals* for a *distributed, context-centric scenario*. The business goals ($BG_1, \ldots, BG_5$) are summarized and briefly discussed in Table 1.

Table 1. Business goals of a distributed, context-centric e-business scenario.

#	Business Goal	Description
BG_1	Commercial transactions of complex products	Enabling transactions of complex products that fulfill a user-defined context is seen as a core value proposition of a context-centric e-business scenario. For buyers, it means a reduction of transaction costs, less effort in making informed buying decisions, and for providers, gaining higher visibility of their offers related to the user-defined context.
BG_2	Decentralization and scalability	The context-centric e-business scenario should be decentralized in order to alleviate the effects of the growing power of mega-platforms and increase the flexibility of integration for a wider range of different product/service domains.
BG_3	Simplicity of use, integration and management	Existing technologies have to be utilized as a way to obviate the aforementioned adverse selection effects. Therefore, a higher level of automatization has to be reached by providing tools and services for modeling contextual information. For example, these can be well-defined, context-aware service interfaces, ranking possibilities of proposals on the buyer side, as well as dynamic service composition and analytics possibilities on the provider side.
BG_4	Enabling trade in any business domain	Since, to our understanding, complex products can encompass any combination of products (*i.e.*, goods) and/or services, the context-centric business environment has to allow trading of any kind of tradable goods. According to [8], these can be categorized as physical goods (*i.e.*, tangible goods that are exchanged on a per piece-basis), digital goods (*i.e.*, intangible goods that are exchanged electronically), services (*i.e.*, activities that one peer performs for the other), as well as financial goods (*i.e.*, amounts of money that are transferred between peers). That implies that the underlying value architecture of a context-centric e-business scenario has to be open for any business domain and has to allow cross-domain transactions.
BG_5	Integration of peer economy principles	In order to support the concept of the peer economy (*i.e.*, peer-to-peer or point-to-point economy [3]), an e-business scenario has to consider the three market design issues, *search, pricing* and *trust*, among trading peers [16]. Therefore, a context-centric e-business scenario has to be designed for an effective matching of sellers and buyers while keeping transaction costs low and for an integration of different pricing models. This is considered important particularly for small and medium-sized sellers (SMEs): on the one hand, it allows SMEs to compete with traditional providers of products/services, and on the other, to gain better utilization of price flexibility. Trust and security are also seen as very important, as they often imply the accuracy of providing information and the reliability of the transaction. One important aspect, given the nature of the provided user-context information, is not only to ensure that data are only used for the purpose the user has agreed to, but also to ensure the privacy of context information and the avoidance of unwanted profiling of individuals.

3.2. Requirements

Having formulated the main business goals ($BG_1, \ldots, BG_5$) of a distributed, context-centric e-business scenario, we will use them as the rationale for defining the main business functionalities (*i.e.*, requirements) that need to be fulfilled on the operational level. To this effect, we will outline the most relevant functional aspects of the e-business model required to be supported by the value architecture.

The value architecture can be described by taking several different points of view. In order to capture specific properties that are relevant for the e-business model, we take the functional viewpoint and focus on architecture as a conceptual structure of an automated software-system. As such, the term of value architecture coincides with the concept of software-system architecture applied in the context of e-business models.

The following functional requirements have been identified as the most relevant requirements for a software-system architecture and can thus be seen as an enabler of a distributed, context-centric e-business model. These are shown in Table 2. On the left, business goals (BG_1, ..., BG_5) are used as rationales for defining requirements (R_1, ..., R_8), followed by short functional descriptions on the right.

Table 2. Requirements for a software-architecture as an underlying value architecture of a distributed, context-centric e-business model.

Rationale	#	Requirement	Description
BG_1, BG_3	R_1	Native support of complex products	Capability for processing an arbitrary combination of individual products and/or services that need to fulfill particular conditions determined by a user's context.
BG_1, BG_3	R_2	Integration of distributed context information	Capability of considering the user's context, which can encompass much more information than those related to the decision criteria and include the wider range of relevant information.
BG_2, BG_3	R_3	Distributed transactions	Possibility of trading products/services from different suppliers, which are available on one or more marketplaces, in a single enclosing transaction from the user's point of view.
BG_4	R_4	Cross-domain transactions	Possibility of trading products/services from different business-domains in a single enclosing transaction.
BG_1, BG_3, BG_5	R_5	Advanced matching	Capability of matching a large number of fragmented, heterogeneous buyers and sellers effectively, keeping transaction costs low.
BG_1, BG_3, BG_5	R_6	Advanced ranking	Supporting buyers in making informed decisions by applying a ranking mechanism that considers context-related constraints to create the "best-fit" list of offers.
BG_5	R_7	Sophisticated reputation and feedback mechanism	Supporting buyers in making informed decisions by considering sophisticated information about potential trading partners.
BG_5	R_8	Flexible price models	Supporting suppliers gaining better utilization or price flexibility.

4. Related Work

In the following, we provide an overview of the operational solutions and concepts regarding the exchange of products and services in commercial terms and compare them to previously-defined requirements (R_1, ..., R_8). We focus our discussion on two main areas. In Section 4.1, we discuss electronic marketplaces as established solutions for commercial exchange, as well as relevant initiatives in that area related to our requirements. In Section 4.2, we elaborate on the smart city as an ICT-based exchange environment and discuss recent smart city architectural approaches.

4.1. Electronic Marketplaces and Related Initiatives

The main economic benefits of electronic marketplaces are to increase market transparency (e.g., to match supply and demand), lower transaction costs (e.g., the buyer's costs of acquiring information about seller prices and product offerings) and coordination costs (e.g., costs related to transaction settlement) [17]. As a market-oriented organization of trading, e-marketplaces support operational activities around sales by matching of supply and demand supported by diverse ranking, feedback and customer review mechanisms. Most of the well-established e-marketplaces operate as silos of information, focusing on the availability of individual products and services.

Generally, e-marketplaces allow transactions within their own boundaries (*i.e.*, particular domain, industry, type of products, *etc.*). There are also marketplaces, e.g., Amazon Marketplace or Alibaba, that emphasize the long tail [18] by offering a wider range of products, especially niche products and allowing cross-domain transactions. Regarding the capabilities of supporting complex products by considering user's contextual information (R_1, R_2), most e-marketplaces enable only compositions of individual products or services within their domain boundaries; or they offer pre-defined combinations of them, which are traditionally bought together and determined by recommender systems (e.g., consider holiday planning, including booking a flight, hotel, rental car and guided tour). The contextualization of user's requests is reduced to existing information about products and/or services.

The marketplace solution proposed by [19]—4CaaSt marketplace—addresses contextualization of user's requests to some extent by enabling the modular composition of individual services across different providers. It is a marketplace for trading cloud services, which equally supports the supply and demand side in a transaction of service compositions regarding a wider and dynamic set of parameters (*i.e.*, related to business goals, services level agreements (SLAs), *etc.*). Even though the 4CaaSt marketplace is an advanced e-marketplace solution in terms of contextualization, it is a centralized domain-specific operational solution, and as such, it does not support cross-domain transactions. Hence, e-marketplace solutions are limited in their capability to support transactions of complex products (R_1, R_3) that need to fulfill a specific user-defined context (R_2) and only provide very limited possibilities to combine transactions of different e-marketplaces without switching among them (R_4).

The Linked Open Commerce (LOC) initiative [20] addresses the problem of structuring data for use in e-commerce. LOC is based on GoodRelations [21], and it provides tools to enrich the description of products and services in a domain- and syntax-neutral way. For a representation of an e-commerce scenario, LOC uses the agent-promise-object principle, where an agent represents the person or organization, the object describes the product/service and the promise represents the rich description of the offer. Thereby, LOC enables a rich description of trading entities, but does not support relations and dependencies among them. Thus, the native support of complex products is missing (R_1, R_2).

The Intention Economy (IE), coined by [22], 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, which is also called project vendor relationship management (VRM), is to equip buyers with tools that make them independent in their relationships with vendors and other parties on the supply-side of markets [23]. 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 *intentcasting*. pRFPs are created as persistent computing objects and published in the Kinetix Rule Engine [24]. The process of matching supply and demand matching 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 a matching with potential vendors, they do not support product/service combinations across different domains (R_1, R_2), as well as cross-domain and distributed transactions (R_3, R_4).

Web of Needs (WoN) is a framework that, according to the authors [25], should serve as a foundation for a distributed and decentralized e-marketplace on top of the Web. The main idea of WoN is to standardize the creation of objects (*i.e.*, owner proxies), which describe supply or demand, represent the intention to enter in a transaction, as well as contain information of the owner, needed for conducting the transaction [25,26]. Modeled in Resource Description Framework (RDF) [27], owner-proxies are published on the web by sending them to the so-called WoN nodes, which are distributed and interconnected. WoN also defines protocols for exchanging messages between owner-proxies, which enables independent matching services to connect owner-proxies and identifying of potential transaction partners. 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.

Generally, native support for describing the need for complex products is provided by the WoN framework. In the case of complex products (*i.e.*, compositions), the user always has the possibility to publish atomic entities, wait for responses and manage the composition manually. However, if the user wants the system to process the whole composition, he or she can publish a complex need, waiting for a matching service capable of interpreting his or her complex need. Thus, 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 framework does not go beyond starting the conversation between them. As to [26], there is no obvious support for conducting cross-domain, distributed transactions (R_3, R_4).

4.2. Smart City Architectures

According to [7], documented smart city cases utilize different architectural approaches. Service Oriented Architectures (SOAs) and n-tier architectures are widely applied, while n-tier is preferred by the majority of smart city cases, encompassing the network, content, intelligence and e-service layers. Most architectures focus on a specific domain (*i.e.*, purpose, like transportation, healthcare, education, building, security and public safety [28]) and propose a technological solution to a specific problem related to that domain [29].

Other approaches, e.g., the City Integrated Service Management Platform [30], the Urban Operating System^{TM} [31] and Stack4Things [32], emphasize the concept of the smart city as an infrastructure composed of people and objects connected through ICT (ubiquitous city) as a unified, distributed and real-time control-platform allowing new smart cities to be created, monitored and optimized.

Various smart city projects, e.g., SCRIBE (Smart Cities Reference Information and Behavior Exchange) [33], Smart Santander [34] or City Sense [35], approach the smart city in terms of semantic models based on the data gathered from around the world, using these in different domains for different applications (e.g., environmental and traffic monitoring, availability of parking spaces, resource management, *etc.*). Where the aforementioned approaches focus on data gathering and aggregation, the Multi-Level Smart City Architecture [36] addresses knowledge extraction and the combination of different domains of a smart city (smart health, energy, transportation, administration or smart industry), thus providing high-level context-aware customized services within these domains.

Additionally, Lea and blackstock [37] introduces the *Smart City Hub* as an architecture framework for the smart city. The hub based on the Internet of Things (IoT) is used as the basis for an easy-to-use service access point for the emerging data infrastructures of a city. As for the authors, this field-tested architecture framework focuses on hub integration, rather than the integration of individual city sub-systems, as in [36].

Conclusively, most of the smart city architectures discussed above focus on one particular domain (*i.e.*, transportation, education, healthcare, *etc.*), providing domain-related products and services. Even though there are some approaches (*i.e.*, Multi-Level Smart City Architecture and Smart City Hub) that introduce an advanced ICT-based environment for exchanging information, products and services, these solutions consider a pre-defined combination of relevant domains, thus not supporting cross-domain transactions.

5. Architecture of the Distributed Market Spaces

In view of our requirements (*cf.* Section 3), contemporary solutions for commercial exchanges are mature in supporting the transaction process for individual products and services (thus, fulfilling $R_5, \ldots, R_8$). However, they are limited in supporting users in transactions of complex products that need to fulfill a specific user-defined context ($R_1, \ldots, R_4$). Existing approaches, such as LOC, IE or WoN (*cf.* Section 4), address some of these requirements, but do not represent a comprehensive solution. Either they provide tools that need to be integrated with other solutions to be fully usable or they propose a framework that addresses only some of the requirements.

The presented electronic marketplaces (*cf.* Section 4.1) and smart city use cases (*cf.* Section 4.2) operate as silos of information. Their architectures focus on ICT-based solutions [29] for particular domains (*i.e.*, transportation, healthcare, education, *etc.*) by providing information and services related to these domains. In order to add value to its participants and stakeholders, smart cities need to become an information marketplace [28] that creates and exchanges information, products and services.

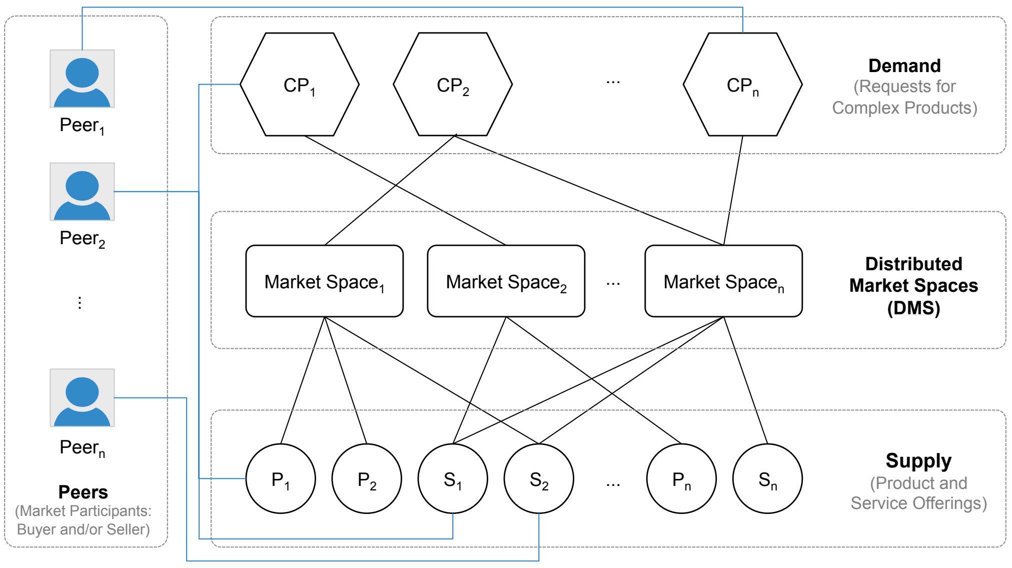
To our knowledge, there is no integrated operational solution that meets all of the aforementioned requirements $R_1, \ldots, R_8$. As part of our vision, we propose our Distributed Market Space (DMS) architecture as an underlying value architecture. As an explanation of our system, we first introduce a high-level overview of the system in Section 5.1 followed by the architecture of the system and its individual components (Section 5.2).

5.1. High-Level Overview

DMS supports market participants in lowering transaction and coordination costs and making transactions of complex products, and it facilitates market exchange by alleviating the effects of the growing power of mega-platforms. In order to achieve this goal, DMS is fully decentralized and allows market places to be instantiated by everybody (e.g., city, company, organization or an individual). It provides inherent support for distributed (buying and selling) transactions, so that buyers can compose arbitrarily complex products easily.

Figure 2 shows a high-level overview of the system, its components and participants. The system is used by market participants (buyers and/or sellers), called *peers*. Peers are potential transaction partners defined by their intention. Buyers are peers intending to buy complex products, while sellers are peers intending to sell products/services. Peers connect to one or more independent *market spaces* (MS). A multitude of these market spaces form the *distributed market space*, where supply meets demand. A peer (in the role of a seller) may offer products and services on one or more market spaces (e.g., $Peer_2$ offers product P_1 and service S_2). In the role of a buyer, a peer may request (complex) products by sending the request to one or more market spaces (e.g., $Peer_2$ also requests the complex product CP_1). In this example, $Peer_1$ acts as a buyer only, $Peer_2$ acts as seller and buyer and $Peer_n$ only sells a service.

As illustrated in Figure 2, *demand* represents requests for complex products as an arbitrary combination of products and/or services that add value for a buyer based on the user's context. Complex products may be requested in a single market space (CP_1 and CP_n) or in multiple market spaces (CP_2). Hence, a buyer may choose to buy the individual components of the complex product on multiple market spaces, and a distributed transaction guarantees that the transaction as a whole is finished or it is canceled.



Peer₁
Peer₂
Peerₙ
Peers
(Market Participants: Buyer and/or Seller)
CP₁
CP₂
CPₙ
Demand
(Requests for Complex Products)
Market Space₁
Market Space₂
Market Spaceₙ
Distributed Market Spaces (DMS)
P₁
P₂
S₁
S₂
Pₙ
Sₙ
Supply
(Product and Service Offerings)

Figure 2. High-level overview of the system, its components and participants.

Supply encompasses sellers offering their products and services in a particular market space (e.g., products P_1 and P_2 are being offered on *Market Space*$_1$ only) or in multiple market spaces (e.g., service S_2 being offered on *Market Space*$_1$ and *Market Space*$_n$).

In the following, Section 5.1.1 describes how a single seller and a single buyer interact in a single market space to complete the transaction for a complex product. Afterwards, Section 5.1.2

extends this use case to include two market spaces with a seller in each market place to show how the distributed transaction works.

5.1.1. Interaction with a Single Market Space Instance

In order to explain how a transaction between a single seller and a single buyer works, this section presents the sequence of steps required to complete such a transaction. Here, we assume the simplest configuration of the DMS where only a single market space instance is involved and the complex product is offered by a single seller. Figure 3 shows the corresponding sequence diagram. Before a market space can dispatch a request for complex products to sellers, the sellers must be registered on that particular market space and indicate what types of products/services are offered.

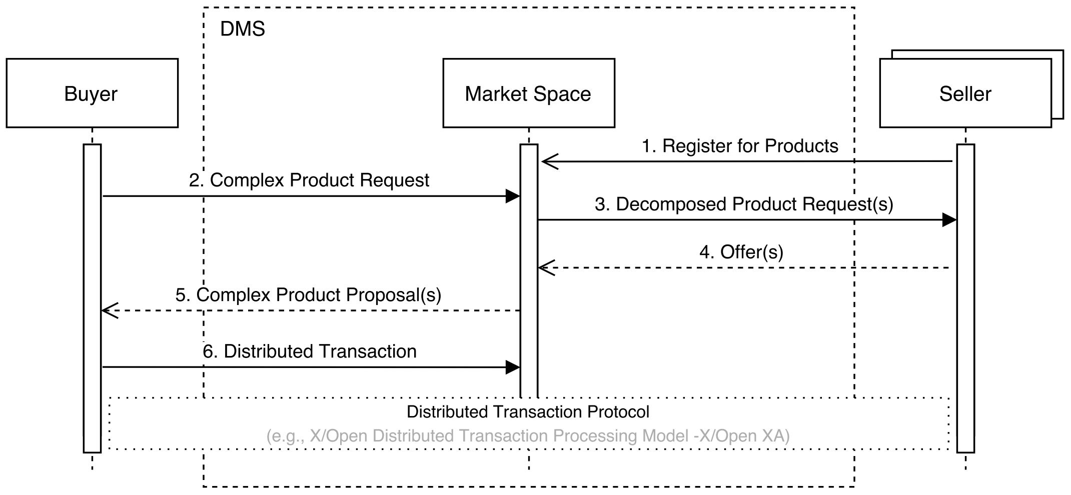

DMS
Buyer
Market Space
Seller
1. Register for Products
2. Complex Product Request
3. Decomposed Product Request(s)
4. Offer(s)
5. Complex Product Proposal(s)
6. Distributed Transaction
Distributed Transaction Protocol
(e.g., X/Open Distributed Transaction Processing Model -X/Open XA)

Figure 3. Transaction of complex product within a single market space.

Thereafter (Step 2), the buyer sends the request for a complex product to the market space. The market space receives the request, decomposes the complex product into individual products/services and creates decomposed product requests and sends them to matching sellers (Step 3). Such a decomposed product request represents a single product/service (as shown in Figure 2 in the "Supply" box). Sellers may choose to send offer(s) to the market space (Step 4), which collects these individual offers and re-combines them into complex product proposals. If multiple complex product proposals exist, the market space may choose to rank them before returning them to the potential buyer (Step 5). The buyer may now select one of the product proposals and initiate a buying transaction (Step 6).

In this example, the "distributed" transaction only involves a single market space. After the transaction is completed, buyer and seller may start the handling of payments, *etc*. For the distributed transaction, we use the existing and well-known model for distributed transaction processing, *i.e.*, distributed transaction processing (DTP), proposed by [38].

5.1.2. Distributed Transaction in the Distributed Market Space

In the second scenario, we take a closer look at the transactions of complex product requests that span across multiple market spaces. In Figure 4, we use two market spaces for the reason of simplicity, but this example can be extended to an arbitrary number of market spaces. In this example, the complex product request is comprised of product P_1 and service S_1. *Seller A* is registered in *Market Space A* as a provider of products of type P_1 (Step 1.1), and *Seller B* is registered in *Market Space B* as a provider of services of type S_1 (step 1.2).

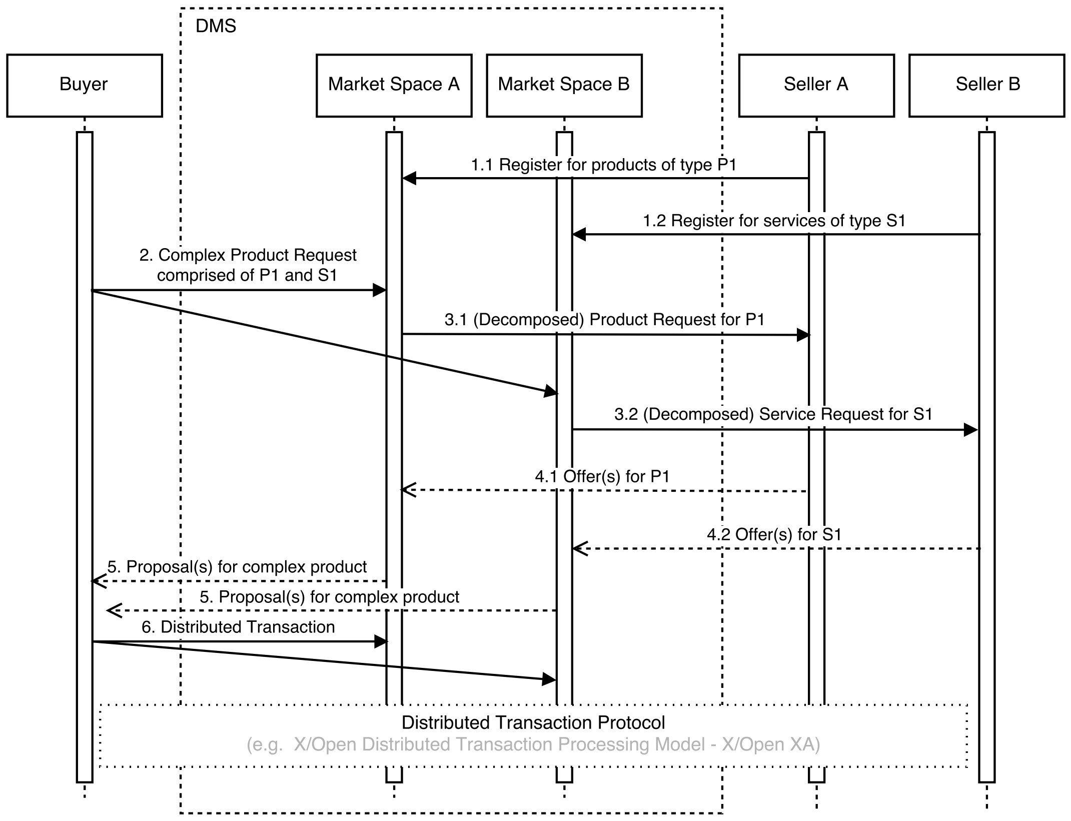

DMS
Buyer
Market Space A
Market Space B
Seller A
Seller B
1.1 Register for products of type P1
1.2 Register for services of type S1
2. Complex Product Request comprised of P1 and S1
3.1 (Decomposed) Product Request for P1
3.2 (Decomposed) Service Request for S1
4.1 Offer(s) for P1
4.2 Offer(s) for S1
5. Proposal(s) for complex product
5. Proposal(s) for complex product
6. Distributed Transaction
Distributed Transaction Protocol
(e.g. X/Open Distributed Transaction Processing Model - X/Open XA)

Figure 4. Distributed transaction of complex products in two market spaces.

The buyer sends a request for the complex product (comprised of P_1 and S_1) to the DMS (*i.e.*, to multiple markets spaces). Each market space decomposes the complex product request and forwards the decomposed product/service requests to matching registered sellers. In this example, *Market Space A* forwards product request P_1 to *Seller A* (Step 3.1). *Market Space B* forwards service S_1 to *Seller B* (Step 3.2). Just like in the previous example (*cf.* Section 5.1.1), sellers make their offers and send them to corresponding market spaces (Steps 4.1 and 4.2). The buyer receives all offers from all involved market spaces (Step 5). As the individual offers do not contain a single and unified offer for all parts of the request, he or she needs to start a distributed transaction that involves both partial offers and start the buying transaction.

As a result, the buyer either receives complex product proposals from one or more market spaces and can directly choose and buy one or he or she needs to compile the complex product himself or herself based on his or her personal preferences. In both cases, the distributed transaction takes place, and the buyer receives the desired complex product (if the transaction succeeds) or does not buy anything at all.

5.1.3. Discovery of DMS Instances

One of the central design goals of the DMS is that there exists no central instance controlling the overall system. Instead, it is fully decentralized and allows DMS instances to be created and operated by everybody. For buyers, it is key to discover those DMS instances that offer products/services matching their product requests. As shown in Figure 3, sellers register at one or more particular DMS instances and indicate the types of products/services that are offered. This information, which is encoded in RDF documents, can be used to identify these DMS instances that offer the requested products/services.

Following the principle of decentralization, we propose to use a decentralized peer-to-peer-based (P2P) discovery mechanism using a distributed hash table (DHT). The idea is that each DMS instance joins the P2P network and publishes a self-description, as well as an aggregated description of all offered types of products/services. This provides inherent scalability properties as an increasing number of DMS instances automatically provides more resources in the P2P network.

We have implemented the discovery service using an existing distributed RDF database called *DecentSparql* [39], which is available as an open source project. It is a Semantic Web database management system with a P2P system to provide full SPARQL 1.1 support, scalability, decentralization, fault-tolerance and infrastructure-free operation. Its architecture is shown in Figure 5. Towards the application that uses DecentSparql, it exposes a standard SPARQL interface using the LUPOSDATE [40] open-source Semantic Web database management system implemented in Java. It supports SPARQL 1.1, ontology languages (such as RDF Schema [41]) and the rule language Rule Interchange Format Basic Logic Dialect (RIF BLD [42]). DecentSparql implements a custom storage backend of LUPOSDATE and, hence, has full access to its internal data structures and optimization capabilities. This storage backend is responsible for the efficient distribution of RDF data and SPARQL queries in the P2P network using a number of distribution strategies available as plug-ins. Finally, DecentSparql interfaces with a number of existing DHT-based P2P implementations, such as TomP2P [43] or Chordless [44].

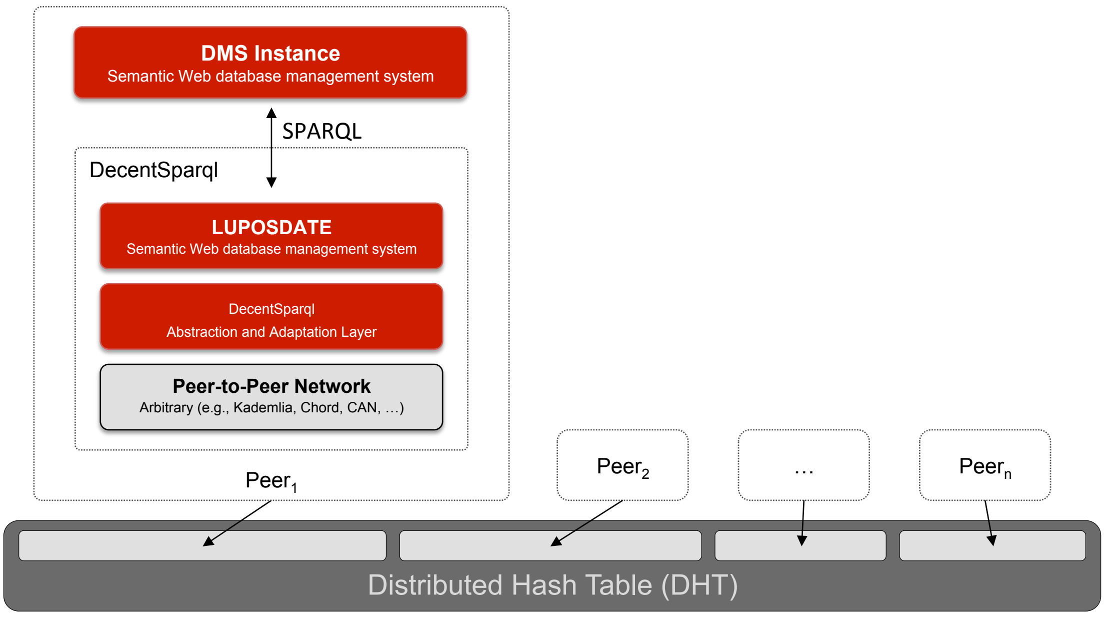

DMS Instance
Semantic Web database management system
SPARQL
DecentSparql
LUPOSDATE
Semantic Web database management system
DecentSparql
Abstraction and Adaptation Layer
Peer-to-Peer Network
Arbitrary (e.g., Kademlia, Chord, CAN, …)
Peer$_1$
Peer$_2$
…
Peer$_n$
Distributed Hash Table (DHT)

Figure 5. Architecture of DecentSparql (adapted from [39]).

As shown in Figure 5, each DMS instance joins the P2P network. However, the distribution of the RDF data and SPARQL queries is transparent for the DMS, as DecentSparql's interface exposes a standard SPARQL interface. Each DMS instance therefore basically stores the registrations of sellers in this database. As each instance performs the same operation, the distributed RDF database ultimately contains all offered products/services along with their respective DMS instances.

For users of the DMS, this requires that they know at least a single DMS instance in order to send queries to the P2P network. A query for a specific product/service sent to a single DMS instance returns a list of all DMS instances that provide this product/service, since the result is transparently compiled from the distributed data store using DecentSparql. The client application can then forward the complex product request to these market spaces that are part of the result set.

5.2. Architecture

After Section 5.1 provided a high-level introduction of the high-level overview of the system, its components and participants, this section provides a more detailed description of the inner workings of the peers and the market instances. It describes the main functional elements of the DMS, their responsibilities and primary interactions in Section 5.2.1, the transaction process and exchanged messages in Section 5.2.2, as well as the prototypical implementation of the aforementioned use case in Section 5.2.3.

5.2.1. Functional View of the DMS

As shown in Figure 6, the peer component is split into two parts: a seller and a buyer side. It can be seen that on the *buyer side*, the tasks of the peer component are:

- to transform a user's intention into a complex product request,
- to distribute these requests to multiple market spaces,
- to receive (partial) complex product offers,
- to re-combine them into multiple complete complex product proposals,
- to rank them according to the buyer's context and requirements and
- to coordinate the distributed buying transaction.

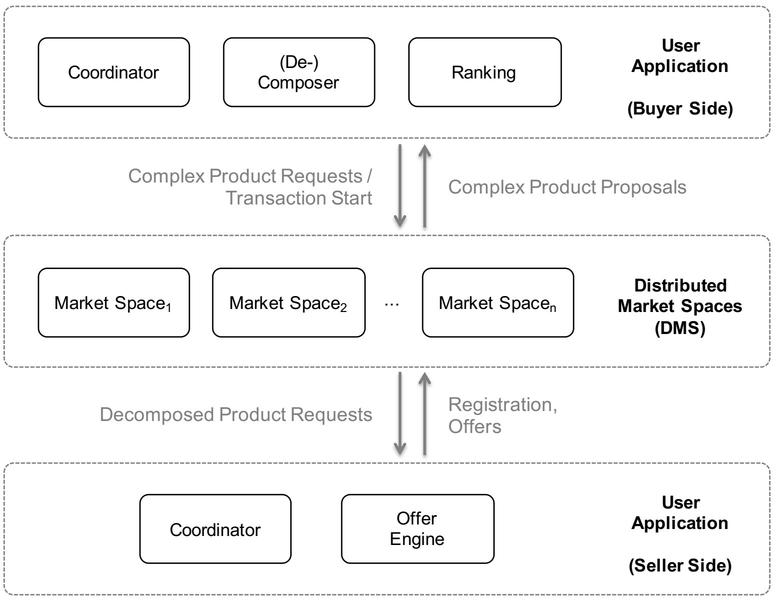

Coordinator
(De-) Composer
Ranking
User Application (Buyer Side)
Complex Product Requests / Transaction Start
Complex Product Proposals
Market Space$_1$
Market Space$_2$
···
Market Space$_n$
Distributed Market Spaces (DMS)
Decomposed Product Requests
Registration, Offers
Coordinator
Offer Engine
User Application (Seller Side)

Figure 6. Functional structure of the peer application.

The buyer side is therefore comprised of the coordinator, the (de-)composer and a ranking component. To describe the complex product request in a domain-agnostic manner, we make use of the well-known Resource Description Framework (RDF) [27], as it is the predominant technique for *machine-readable representations of knowledge* on the web. This allows for the implementation of the peer application in a completely product- and service-agnostic way, so that it can be used for any type of application.

In addition, it enables specifying complex product requests that are comprised of products/services from very different domains. This immediately allows us to integrate an enormous body of existing vocabulary (*i.e.*, ontologies) and world knowledge (*i.e.*, existing machine readable data on the web, such as e.g., DBPedia).

The *(de-)composer* breaks up complex product requests described in RDF into individual product/service requests and communicates them to the DMS. Upon receiving individual proposals, it re-combines them into multiple complete complex product proposals and passes them to the ranking engine. Given a set P of proposals, the ranking engine can basically be represented as an assignment function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, assigning a value $v_p \in \mathbb{R}$ to a given vector $\mathbf{x}_p = (x_1, \ldots x_n), x_i \in \mathbb{R}$, which in turn represents a single proposal $p \in P$. Therefore, f can be seen as an objective function, which rates the proposals to a specific order based on its value (*i.e.*, $f(\mathbf{x}_i) < f(\mathbf{x}_j) \rightarrow p_i \in P$ matches better than $p_j \in P$). As a consequence, the smallest value of f computed over P denotes the best proposal according to the objective function. It must be noted that even though the terminology is similar, we actually do not cope with the optimization problem due to the impossibility to alter the single values $x_i \in \mathbf{x}$ directly; thus, the function is only used for ranking purposes.

User preferences can be expressed as constraints to the objective function. However, since such constraints divide the search space into feasible and infeasible regions, they can be seen as a kind of filter, which removes proposals from the search space and as such does not fit the concept of ranking. Therefore, solving a constrained problem is substituted by solving an unconstrained problem by getting rid of the constraints in favor of a penalty function $P(\mathbf{x})$ [45,46], which is added to the objective function's value. A new unconstrained objective function F is created, such that $F(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x})$. The result of $P(\mathbf{x})$ increases when user-defined preferences are violated. P can now, e.g., be utilized to express relationships between proposal properties (e.g., perfect substitutes, *etc.*) and other preferences (e.g., quality-over-price measures). In addition, data from the Internet of Things can be integrated in the penalty function to include user context data (e.g., using approaches, such as [47,48]), which uses the user's context (e.g., the current weather) or other IoT sources of data to calculate the penalty for the proposals, so that the best matching proposal is assigned the lowest penalty value. The ranking engine can therefore now assign an objective function value to each given proposal and order them by their objective function value. Other techniques for getting rid of constraints are also feasible and may incorporate methods of repairing infeasible proposals/solutions [49]. Such approaches may be evaluated in the future.

On the *seller side*, the peer component's tasks are:

- to register with the offered products/services,
- to receive decomposed product requests and
- to provide offers.

On the seller side, the core component is the offer engine, which provides a price tag for each product/service request and sends offers back to the market space. Additionally, the offer engine can encompass further functionalities depending on, *i.e.*, the type and size of the seller (e.g., individuals, institutions or companies), the type of products/services they offer and other seller-related characteristics, which is out of scope of this paper.

The coordinator component is responsible for supervising the whole process and the distributed (buying) transaction. By using the aforementioned model for distributed transaction processing (*cf.* Section 5.1.1), the coordinator draws on the transaction management process, assigning identifiers to transactions, monitoring their progress and taking responsibility for transaction completion, as well as for failure recovery and rollback.

The functional structure of the market space is depicted in Figure 7. The market space re-uses components for de-composition and ranking from the peer application and adds components for matching supply and demand, as well as for the registration of sellers and their offerings.

The matcher component includes the matching service and the underlying domain-agnostic database. The database contains information about registered sellers, as well as the description of available products and services they can potentially offer. Again, these data are encoded in RDF so that the matcher and registration component are independent from the actual product's domain. However, a market space that specializes, e.g., in tourism may choose to bring in domain knowledge in the form of, e.g., OWL ontologies or sensor data from a city, such that the matching result is improved.

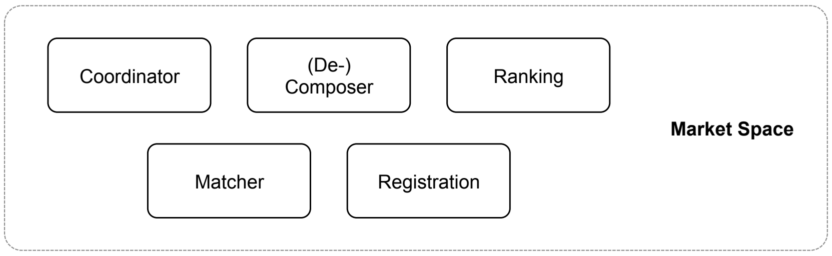


Figure 7. Functional structure of the market space.

5.2.2. Information View of the DMS

After describing the main functional components of the DMS in the following, we outline the transaction process and provide more details about the information flow and exchanged messages between functional components.

The transaction process shown in Figure 8 is modeled using the Business Process Modeling Notation 2.0 (BPMN 2.0 [50]). It outlines the main activities within the transaction process, taking the user application perspective, and shows some message exchanges. In the following and as a matter of example, we describe a product/service offering (PSO) message (highlighted in blue) and a decomposed complex product request (PSD) message (highlighted in green).

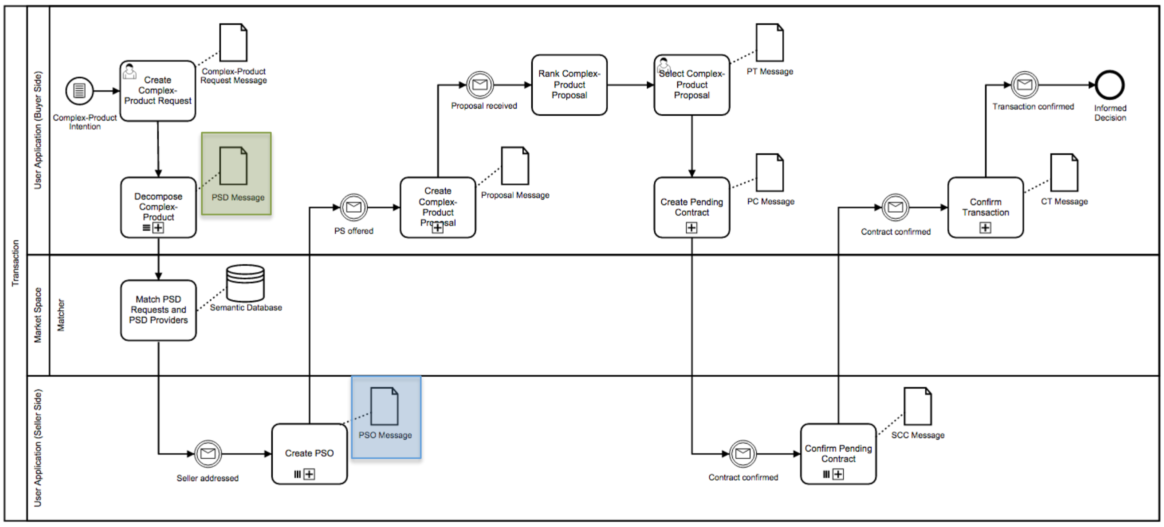


Figure 8. Transaction process of complex products (excerpt).

The messages exchanged in the DMS are RDF documents. In principle, such documents may have arbitrary content, and we could have devised our own proprietary ontology to describe the products/services of our use case. However, as it is good practice in the Semantic Web to reuse existing vocabularies to improve interoperability, we have used existing vocabularies. In the following, we show examples of such messages for the use case described in Section 2.3, where the user is a tourist planning a theater evening. Due to space constraints, we concentrate on the theater ticket.

As a base for the message description, we use the existing good relation ontology [21] and, in particular, its domain vocabulary for event tickets. Listing 1 presents a part of an RDF description of the PSO message in Turtle syntax (Terse RDF Triple Language [51]). The triples state that a particular event is offered by the company *TRIO Tickets* and that it is accessible through a particular URI (Uniform Resource Identifier) (Lines 8–11), followed by triples stating details about the offering, e.g., name, description and further price specification (Lines 12–21). The corresponding graph of the PSO message is shown in Figure 9.

Listing 1: Exemplary description of a PSO message.

```
 1
 2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
 4 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
 5 PREFIX gr: <http://purl.org/goodrelations/v1#>
 6 PREFIX tio: <http://purl.org/tio/ns#>
 7 PREFIX dms: <http://www.itm.uni-luebeck.de/dms/#>
 8
 9 dms:ticket1 a tio:TicketPlaceholder ;
10 rdfs:label "Ticket for Chicago Musical at Alte Oper Frankfurt"@en ;
11 tio:accessTo <http://data.linkedevents.org/event/chicagomusical>.
12 dms:TRIO Tickets ltd. gr:offers dms:PSO1.
13 dms:PSO1 a gr:Offering ;
14 gr:name "Ticket for Chicago Musical"@en ;
15 gr:description "The #1 American Musical in Broadway History:Chicago at Alte Oper Frankfurt"@en ;
16 gr:includes dms:ticket1 ;
17 gr:hasBusinessFunction gr:Sell ;
18 gr:hasPriceSpecification
19         [ a gr:UnitPriceSpecification ;
20         gr:hasCurrency "USD"@en ;
21         gr:hasCurrencyValue "49.50"^^xsd:float ;
22         gr:validThrough "2016-09-26T23:59:59"^^xsd:dateTime ].
```

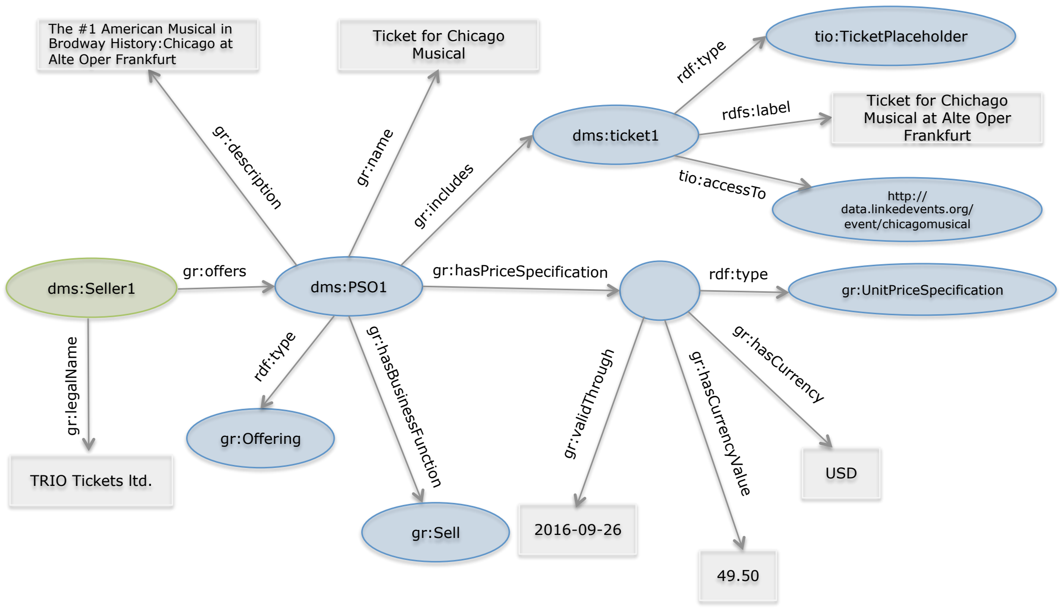


Figure 9. Resulting graph of PSO message from Listing 1.

The PSD message on the other hand is described as a SPARQL query [52] and shown in Listing 2.

5.2.3. Demonstrator Implementation

Currently, we have implemented a prototype of our proposed architecture that operates in a headless manner, *i.e.*, we manually craft the complex product request and submit it to the DMS.

The DMS has been implemented in the Java programming language using a standard message queueing system (Apache ActiveMQ [53]), and the peer applications interact with the DMS by exchanging messages via the message queueing system. The implementation of our prototype is already able to handle the aforementioned use case.

The ranking component is implemented using a very simple objective function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ for demonstration purposes where x_1 denotes the price of a theater ticket, x_2 denotes the price of the Italian restaurant and x_3 denotes the distance in meters needed to walk between the stages (which can be easily calculated by geodata information). The function itself is mainly cost-based and can be expressed as $f(\mathbf{x}) = (x_1 + x_2) * x_3$. The penalty function now uses the weather forecast (which is gathered online from the Internet) to weight the distance x_3. The distance impact on the overall function can be expressed as in example 1.

Example 1.

$$P(\mathbf{x}) = \begin{cases} 0.1 * x_3 & \text{if temperature forecast} > 25\,^{\circ}C \\ 10 * x_3 & \text{if temperature forecast} \geqslant 15\,^{\circ}C \wedge \ \leqslant 25\,^{\circ}C \\ 100 * x_3 & \text{else} \end{cases}$$

such that distances loose impact according to the weather forecast. The weather impact given in this example is quite excessive due to demonstration purposes and can be adapted to other values. The overall function $F(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x})$ now assigns values to proposals according to the overall objective (price sensitiveness), but still taking user preferences like weather conditions into account. More sophisticated objective and penalty functions can be used to model a more complex ranking behavior and to provide best matches according to user preferences.

Listing 2: Example of a PSD message as a SPARQL query.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX tio: <http://purl.org/tio/ns#>
PREFIX dms: <http://www.itm.uni-luebeck.de/dms/#>

SELECT ?ticket, ?currency, ?price
WHERE
{
 ?ticket a tio:Ticket.
 {
        { ?ticket tio:accessTo <http://data.linkedevents.org/event/chicagomusical>. }
        UNION
        { ?ticket tio:scope ?s.
         ?s tio:accessTo <http://data.linkedevents.org/event/chicagomusical>. }
 }
 ?offer a gr:Offering.
 {
        { ?offer gr:includes ?ticket. }
        UNION
        { ?offer gr:includesObject ?t.
         ?t a gr:TypeAndQuantityNode.
         ?t gr:typeOfGood ?ticket. }
 }
 ?offer gr:hasPriceSpecification ?p.
         ?p a gr:UnitPriceSpecification.
         ?p gr:hasCurrency ?currency.
         ?p gr:hasCurrencyValue ?price.
}
ORDER BY (?price) LIMIT 1000
```

A current limitation of the overall system is that we have not yet implemented the distributed transaction in order to buy parts of a complex product/service from different market spaces. In addition, we still have to implement generic ranking algorithms for different types of user-defined ranking. Furthermore, the use of a DHT-based P2P system for the discovery inherently has the problem of finding an entry point to the P2P system. In our current implementation, we use a well-known address for this. The challenge for the future in making the system usable is to provide use-case specific applications that convert user input automatically into complex product requests.

6. Conclusion and Future Work

In order to identify the main objectives of a distributed, context-centric e-business scenario, we analyzed its business dimension and defined the main business goals. Hence, we formulated the value proposition of the underlying business model: an e-business environment that supports market participants (buyers and sellers) in lowering transaction and coordination costs and in making commercial transactions of complex products and which facilitates the market exchange by alleviating the effects of the growing power of mega-platforms.

Afterwards, we operationalized these business goals into functional requirements that such an e-business model needs to fulfill on the operational level in order to provide for the formulated value proposition. We discussed the existing solutions of commercial exchange environments, evaluated them against our requirements, identified their shortcomings and concluded that neither of them is suited as an underlying architecture to support distributed, context-centric, e-business scenarios in the Internet of Everything.

Consequently, we introduced our Distributed Market Spaces (DMS) architecture as a concept of the underlying value architecture, required to support the operational aspects of the above-mentioned business goals, overcoming the shortcomings of existing solutions. The DMS is context-centric (by providing the best matching offers regarding the user's context), highly scalable, strictly decentralized and generic in its nature (not limited to any certain business domain).

Finally, we conducted an initial demonstration of our proposed architecture using an exemplary use case representing the complexity of a smart city e-business scenario.

In our future work, we will concentrate on two areas that we feel require intensive further work: firstly, the extensive modeling of contextual information on the user application side, and secondly, on the implementation of the prototype to conduct a sophisticated analysis of the strengths and weaknesses of the proposed value architecture.

Acknowledgments: This work is kindly supported by Baden-Wuerttemberg Cooperative State University Mannheim, Department of Business Informatics.

Author Contributions: Dennis Pfisterer and Mirjana Radonjic-Simic worked on the high-level overview and functional structure of the DMS architecture as well as on the demonstrator implementation. Mirjana Radonjic-Simic worked on the overall objectives of a distributed, context-centric e-business scenario, provided the overview of the relevant state of the art and contributed on the information view of the DMS architecture. Julian Reichwald worked on the definition of a smart city context as well as on the ranking method based on unconstrained objective functions using penalty mechanisms.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco. The Internet of Everything for Cities. 2013. Available online: http://www.cisco.com/web/about/ac79/docs/ps/motm/IoE-Smart-City_PoV.pdf (accessed on 8 January 2016).
2. Akerlof, G.A. The market for lemons: Quality uncertainty and the market mechanism. *Q. J. Econ.* **1970**, *84*, 488–500.
3. Wang, R. *Disruptive Digital Business*; Harvard Business Review Press: Boston, MA, USA, 2015.
4. Inspiring the Internet of Things, 2011. Available online: http://www.alexandra.dk/uk/services/Publications/Documents/IoT_Comic_Book.pdf (accessed on 8 January 2016).
5. Caragliu, A.; Del Bo, C.; Nijkamp, P. Smart Cities in Europe. In Proceedings of the 3rd International Conference on Regional Science (CERS), Kosice, Slovakia, 7–9 October 2009.
6. Komninos, N.; Pallot, M.; Schaffers, H. Special Issue on Smart Cities and the Future Internet in Europe. *J. Knowl. Econ.* **2013**, *4*, 119–134.
7. Anthopolous, L.; Fitsilis, P. Exploring Architectural and Organizational Features in Smart Cities. In Proceedings of the 16th International Conference on Advanced Communication Technology, Pyeongchang, Korea, 16–19 February 2014.

8. Grefen, P. *Beyond E-Business: Towards Networked Structures*; Routledge: New York, NY, USA, 2016; pp. 53–246.
9. Teece, D.J. Business models, business strategy and innovation. *Long Range Plan.* **2010**, *43*, 172–194.
10. Osterwalder, A.; Pigneur, Y. An eBusiness model ontology for modeling eBusiness. In Proceedings of the 15th Bled Electronic Commerce Conference eReality: Constructing the eEconomy, Bled, Slovenia, 17–19 June 2002.
11. Stähler, P. Business models as an unit of analysis for strategizing. In Proceedings of the International Workshop on Business Models, Lausanne, Switzerland, 4–5 October 2002; pp. 4–5.
12. Osterwalder, A.; Pigneur, Y. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Thallengers*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
13. Gassmann, O.; Frankenberger, K.; Csik, M. *The Business Model Navigator: 55 Models that will Revolutionise Your Business*; Pearson: London, UK, 2014; pp. 3–14.
14. Amit, R.; Zott, C. *Value Creation in E-Business*; John Wiley & Sons, Ltd.: Malden, MA, USA, 2001; Vol. 22, pp. 493–520.
15. Rozanski, N.; Woods, E. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*; Addison-Wesley: Boston, MA, USA, 2011; pp. 12–29.
16. Einav, L.; Farronato, C.; Levin, J. Peer-to-Peer Markets, Technical Report; National Bureau of Economic Research: Cambridge, MA, USA, 2015.
17. Bakos, Y. The emerging role of electronic marketplaces on the Internet. *Commun. ACM* **1998**, *41*, 35–42.
18. Anderson, C. *The Long Tail*; Hyperion: New York, NY, USA, 2006.
19. García-Gómez, S.; Jimenez-Ganan, M.; Taher, Y.; Momm, C.; Junker, F.; Biro, J.; Menychtas, A.; Andrikopoulos, V.; Strauch, S. Challenges for the comprehensive management of Cloud Services in a PaaS framework. *Scalable Computing Pract. Exp.* **2012**, *13*, 201–214.
20. Kingsley, I.; Hepp, M.; Bucchi, A. Linked Open Commerce. 2015. Available online: http://linkedopencommerce.com (accessed on 8 January 2016).
21. Hepp, M. Goodrelations: An Ontology for Describing Products and Services Offers on the Web. In *Knowledge Engineering: Practice and Patterns*; Springer: Heidelberg, Germany, 2008; pp. 329–346.
22. Searls, D. *The Intention Economy: When Customers Take Charge*; Harvard Business Press: Boston, MA, USA, 2013.
23. Berkman Center for Internet and Society at Harvard University. ProjectVRM. 2015. Available online: http://blogs.law.harvard.edu/vrm/projects/ (accessed on 8 January 2016).
24. Berkman Center for Internet and Society at Harvard University. ProjectVRM Wiki. 2015. Available online: http://cyber.law.harvard.edu/projectvrm/Main_Page (accessed on 8 January 2016).
25. Kleedorfer, F.; Busch, C.M.; Pichler, C.; Huemer, C. The Case for the Web of Needs. In Proceedings of the 2014 IEEE 16th Conference on Business Informatics (CBI), Geneva, Switzerland, 14–17 July 2014; Volume 1, pp. 94–101.
26. Kleedorfer, F.; Busch, C.M. Beyond Data: Building a Web of Needs. In Proceedings of the WWW2013 Workshop on Linked Data on the Web, Rio de Janeiro, Brazil, 14 May 2013.
27. W3C. Resource Description Framework (RDF), 2015. Available online: http://www.w3.org/RDF/ (accessed on 8 January 2016).
28. Holler, J.; Tsiatsis, V.; Mulligan, C.; Avesand, S.; Karnouskos, S.; Boyle, D. *From Machine-to-machine to the Internet of Things: Introduction to a New Age of Intelligence*; Academic Press: Oxford, UK, 2014.
29. da Silva, W.M.; Alvaro, A.; Tomas, G.H.; Afonso, R.A.; Dias, K.L.; Garcia, V.C. Smart cities software architectures: a survey. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal, 18–22 March 2013; pp. 1722–1727.
30. Lee, J.; Baik, S.; Lee, C. Building an integrated service management platform for ubiquitous cities. *Computer* **2011**, *44*, 56–63.
31. PlanIT. Introduction to PlanIT Urban Operating System^{TM} Architecture, 2015. Available online: http://living-planit.com/pdf/living-planit-introduction-to-uos-architecture-whitepaper-2015-05-24-v14.pdf (accessed on 8 January 2016).
32. Merlino, G.; Bruneo, D.; Distefano, S.; Longo, F.; Puliafito, A. Stack4Things: Integrating IoT with OpenStack in a Smart City Context. In Proceedings of the 2014 International Conference on Smart Computing Workshops (SMARTCOMP Workshops), Hong Kong , China, 5 November 2014; pp. 21–28.

33. IBM Research. SCRIBE Models and Methodology: Smart Cities Reference Information and Behavior Exchange Ontologies, 2015. Available online: http://researcher.watson.ibm.com/researcher/view_group.php?id=2505 (accessed on 8 January 2016).
34. SmartSantander. SmartSantander Project. 2015. Available online: http://www.smartsantander.eu (accessed on 8 January 2016).
35. CitySenseProject. City Sense Project. 2015. Available online: http://www.citi-sense.eu (accessed on 8 January 2016).
36. Gaur, A.; Scotney, B.; Parr, G.; McClean, S. Smart City Architecture and its Applications Based on IoT. *Proced. Comput. Sci.* **2015**, *52*, 1089–1094.
37. Lea, R.; Blackstock, M. Smart Cities: An IoT-centric approach. In Proceedings of the Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing, Saint Etienne, France, 1–2 September 2014; pp. 1–2.
38. X/Open Company Limited. Distributed Transaction Processing: The XA Specification. Technical Report. Available online: http://pubs.opengroup.org/onlinepubs/009680699/toc.pdf (accessed on 8 January 2016).
39. Mietz, R.; Groppe, S.; Oliver Kleine, D.B.; Fischer, S.; Römer, K.; Pfisterer, D. A P2P Semantic Query Framework for the Internet of Things. *PIK–Praxis Inform. Kommun.* **2013**, *36*, 73–79.
40. Groppe, S. *Data Management and Query Processing in Semantic Web Databases*; Springer Verlag: Heidelberg, Germany, 2011.
41. RDF Schema. Available online: https://www.w3.org/TR/rdf-schema/ (accessed on 23 March 2016).
42. RIF BLD. Available online: https://www.w3.org/TR/rif-bld/ (accessed on 23 March 2016).
43. Bocek, T. TomP2P, a P2P-Based Key-Value Pair Storage Library. Available online: http://tomp2p.net/ (accessed on 23 March 2016).
44. Kihlgren, M. Chordless|Free System Administration Software Downloads at SourceForge.net. Available online: http://sourceforge.net/projects/chordless/ (accessed on 23 March 2016).
45. Vanderplaats, G. *Numerical Optimization Techniques for Engineering Design—With Applications*; McGraw-Hil: New York, NY, USA, 1984.
46. Bomze, I.; Grossmann, W. *Optimierung—Theorie und Algorithmen*; BI Wissenschaftsverlag: Mannheim, Germany, 1993.
47. Mietz, R.; Groppe, S.; Römer, K.; Pfisterer, D. Semantic Models for Scalable Search in the Internet of Things. *J. Sens. Actuator Netw.* **2013**, *2*, 172–195.
48. Pfisterer, D.; Römer, K.; Bimschas, D.; Kleine, O.; Mietz, R.; Truong, C.; Hasemann, H.; Kröller, A.; Pagel, M.; Hauswirth, M.; *et al.* SPITFIRE: Toward a Semantic Web of Things. *IEEE Commun. Mag.* **2011**, *49*, 40–48.
49. Wallace, R. *Analysis of Heuristic Methods for Partial Constraint Satisfaction Problems*; Springer: New York, NY, USA, 1996, 482–496
50. OMG. Business Process Model and Notation Version 2.0. 2015. Available online: http://www.omg.org/spec/BPMN/2.0/ (accessed on 4 March 2016).
51. W3C. RDF 1.1 Turtle. 2014. Available online: http://www.w3.org/TR/turtle/ (accessed on 4 March 2016).
52. W3C. SPARQL Query Language for RDF. 2015. Available online: http://www.w3.org/TR/rdf-sparql-protocol/ (accessed on 4 March 2016).
53. Apache. Apache ActiveMQ. 2015. Available online: http://activemq.apache.org/ (accessed on 4 March 2016).