Journal of Universal Computer Science, vol. 18, no. 17 (2012), 2452-2473
submitted: 16/1/12, accepted: 29/8/12, appeared: 1/9/12 © J.UCS

A Conceptual Model for IT Service Systems

Ajantha Dahanayake
(Prince Sultan University, Riyadh, Kingdom of Saudi Arabia
adahanayake@pscw.psu.edu.sa)

Bernhard Thalheim
(Christian-Albrechts University of Kiel, Kiel, Germany
thalheim@is.informatic.uni-kiel.de)

Abstract: Although services are developed, used, applied and intensively discussed in nowadays IT practice, the concept of an IT service has not yet been introduced. Services are IT artifacts that can be used by many users in different context at different points of time in different locations and serve a certain purpose. They provide the data and functionality at the best point of time, in the agreed format and quality for the right user with the right location and context.
We generalize some of the introduced notions such as the REA framework (resource-event-agent) and introduce a framework for conceptual modeling of IT service systems that is based on the classical rhetorical frame introduced by Hermagoras of Temnos (Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W7: Who, what, when, where, why, in what way, by what means)). Services are primarily characterized by W4: wherefore (end), whereof (source), wherewith (supporting means), and worthiness ((surplus) value). Additionally, the purpose can be characterized by answering the why, whereto, when, and for which reason W4 questions. The secondary characterization W14H is given by characterizing user or stakeholder (by whom, to whom, whichever), the application domain (wherein, where, for what, wherefrom, whence, what), the solution they are providing (how, why, whereto, when, for which reason), and the additional context (whereat, whereabout, whither, when).

Keywords: Service Systems, Conceptual Modelling, IT Service Systems, Web-based services, Classical Rhetorical Frame, Hermagoras of Temnos, Conceptual Model for Services
Categories: H.1.1, H.2.3, H.2.4, H.4.2, H.4.3, J.4, J.7

1 Introduction

The growth of the IT service sector in global economics has influenced even larger manufacturing firms to seek dramatic shifts in revenue deriving from IT services [Bryson, 04]. IT services are in the center stage in global business arena, articulating IT service innovation as a means to fuel the economic growth through raising quality and productivity levels [Bosworth, 03]. IT services as knowledge-intensive business services have become drivers for improving business performances [Spohrer, 08]. As a consequence Service Science has become the interdisciplinary academic field that studies service systems engineering [Spohrer, 07a].

An IT service is being defined using different abstraction models with varying applications representing multitude of definitions of the service concept [Goldstein, 02]. The increasing interests in services have introduced service concept's abstraction

into levels such as; business services, web services, software-as-a-service (SaaS), platform-as-a-services and infrastructure-as-a-service [Bergholtz, 10]. Service Architectures are proposed as means to methodically structure systems [Erl, 07], [Arsanjani, 08], [Zimmerman, 04] and [Stojanovic, 04]. The service delivery discussions in research have mainly concentrated on the relationship between an economic activity of increasing the business value and technology as a means; in this case web services have to deliver effective and efficient services. As a consequence the economic activity that supports a business process is defined as a business service. The web service design, development, and orchestration addresses the identification of the right services including the organization of a manageable hierarchy of composite services for choreographing supporting business process [Papazoglou, 06].

However, the service delivery has not paid attention to the innovation, design, and development of an IT artifact as a service [Goldstein, 02]. Research has used the concept of service in IT artifact innovation. An information service [Chen, 09] is an innovation of the conceptual model of an IT.

Although services are developed, used, applied and intensively discussed in practice, the conceptual model of an IT service system has not yet been introduced in a meaningful manner. We introduce a conceptual model for IT service systems and try to fill the gap in research contributions of conceptual models of services. Currently services are either generalized to any service system or specific to web services from a technological abstraction levels.

The service concept plays an increasingly important key role in service design and development literature. Surprisingly little has been written about the service concept itself and its important role in IT service systems innovation, design and development [Goldstein, 02]. The service concept defines the what, how and who on what basis of service innovation, design, and development helps mediate between customer or consumer needs and an organizations strategic intent [Goldstein, 02] . Therefore, when it is extended above the generalized business and technological abstraction levels, the conceptual model for IT service system can serve the following purposes:

- Fundamental elements for developing applications;
- Organizing the discrete functions contained in (business) applications comprised of underlying business process or workflows into inter operable, (standards-based) services;
- Services abstracted from implementations representing natural fundamental building blocks that can synchronize the functional requirements and IT implementations perspective;
- Services to be combined, evolved and/or reused quickly to meet business needs;
- Represent an abstraction level independent of underlying technology.

The above described purposes of a conceptual model for IT service system is illustrated in Figure 1 which extends the SOA stack model [Hirschheim, 10] to IT service systems innovation, design and development.

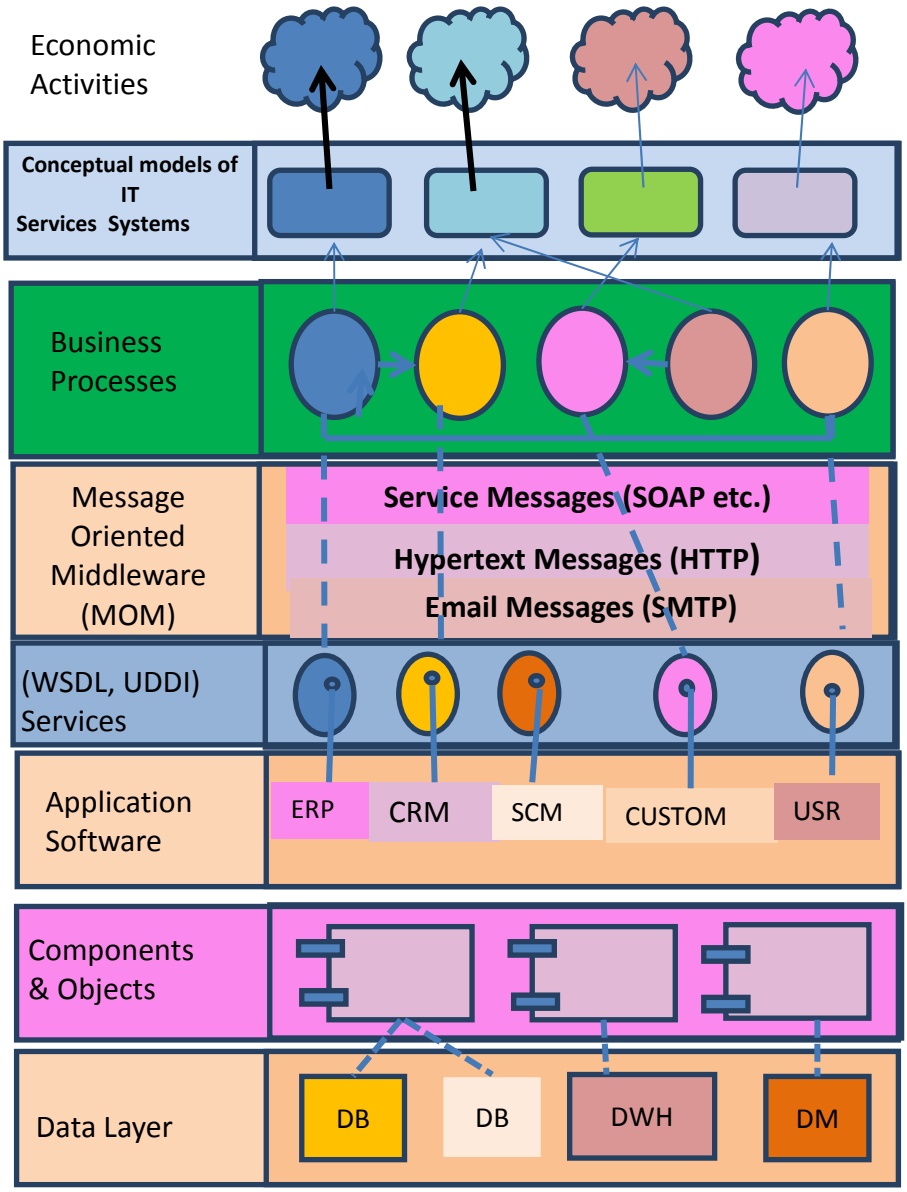

Figure 1: Contribution of conceptual models for IT service system innovation, design and development within the business and technology abstraction layers (extension to the SOA stack model [Hirschheim, 10].

1.1 IT Service Systems

IT Service systems combine and integrate the value created in different design contexts like person-to-person encounters, technology enabled self-service, computational services, multi-channel, multi-device, and location-based and context-aware services [Spohrer, 07a], [Maglio, 06], and [Spohrer, 07b]. There is a substantial subset of IT service systems that can be described as "information-intensive" [Stojanovic, 04], and [Chen, 09] , and it is desirable to take a more abstract view of service contexts that highlights what person-to-person, self-service, and automated or computational services have in common rather than emphasizing their differences [Glushko, 10].

The IT service system view reveals the intrinsic design challenges that derive from the nature of the information required to perform a service, and emphasizes the design choices that allocate the responsibility to provide this information between the service provider and service consumer. Taken together, the information requirements and the division of labor for satisfying them determine the nature and intensity of the interactions in the IT service system. This more abstract approach that applies to all contexts overcomes many of the limitations of design approaches that focus more narrowly on the distinctive concerns of each context [Glushko, 10].

This paper aims at a general notion for a conceptual model for IT service systems: A general definition of a service seems to be rather difficult. We use however a framework that separates concerns such as service as a product, service as an offer, service request, service delivery, service application, service record, service log or archive and also service exception. This separation of concern allows supporting a general characterization of services by their ends, their stakeholders, their application domain, their purpose and their context.

The paper is organized as follows: Section 2 revisits related works and thus highlights the uniqueness of our contribution. Section 3 summarizes the classical rhetorical frame introduced by Hermagoras of Temnos which influenced our philosophical foundation in generalizing some of the concepts found in the REA framework. Section 4 outlines our conceptual model for IT service systems innovation, design and development. We conclude in Section 5 providing a discussion, and future research initiatives.

2 Related Works and Research

2.1 Service Science

Service science is an initiative of IBM [IBM, 04] that launched the emerging academic field for studying service systems to discover underlying principles that can guide the innovation, design and development of service systems [Spohrer, 07a]. As a distinct interdisciplinary field, it searches for an ideology and a unifying paradigm [IBM, 04].

2.2 The REA (Resource-Event-Agent) Ontology

REA ontology's conceptual origin lies in the traditional accounting applications which use the double-entry bookkeeping technique for managing financial systems

where business transactions are recorded as a credit and a debit thus a double entry. REA ontology formulated as in the original article [McCarthy, 82] was further articulated and extended by others, e.g., [Geerts, 99] and [Hurby, 06]. The core concepts in the REA ontology are *resources*, *economic event*, and *agent*. The fundamental behind the ontology is that there are two ways agents can increase or decrease the value of their resources: through exchange and conversion process [Hurby, 06]. An economic resource is a valuable good, right, or service that is at a given point is under the identifiable control of an economic agent. An economic resource is under the control of an economic agent if that person owns the resources or otherwise able to derive economic benefit from it. If two economic agents desire to obtain control over one or more economic resources controlled by the other agent, then both agents may wish to engage as trading partners in an economic exchange, which is a business transaction that transfers the control of resources between agents. A transfer of control of a resource(s) from one agent to another agent is modeled as an economic event in which the concerned resource(s) are identified as a stockflow relation and agents anticipate in provider and receiver roles. Economic reciprocity in exchanges is modeled through the duality relation between economic events and requesting events such as payments, in which the provider and receiver roles of the involved agents are switched.

2.3 The RSS Model

The Resource-Service-Systems (RSS) model for service systems [Poels, 10] is an adaptation of REA model stressing that REA is a conceptual model of economic exchange, and it is not a model of service exchange, because of the influence of Service-Dominant Logic (SDL) [Vargo, 04a] in the RSS. SDL has been proposed as the philosophical foundation of service science for providing the right perspective, vocabulary, and assumptions to build a theory of service science, their configurations and models of interaction [Vargo, 04a]. SDL sees all economic activity as service exchange between service systems [Vargo, 08]. In SDL, service is a competence that exchanges for the benefit of other service systems. In contrast to traditional Goods-Dominant Logic (GDL) model [Vargo, 04b], SDL sees a service as a collaborative process in which each party brings in or makes accessible its unique resources. RSS model serves as a generalized conceptual model for positioning service within the resources and systems, but it does not help in conceptualizing an IT service system at the event of innovation, design and development of IT artifacts.

2.4 The Three Perspectives on Services: Abstraction, Restriction, and Co-Creation

The three perspectives on services: abstraction, restriction, and co-creation introduced as a conceptual model of service concept that views services as perspectives on the use and offering of resources [Bergholtz, 10]. The perspectives addressed by this conceptual model are: service as a means for abstraction; service as means for providing restricted access to resources; and service as a means for co-creation of value. It relies on the argument that: in the classical manufacture economic model service has been defined and characterized by identifying properties such as intangibility, inseparability, heterogeneity, and perishability in the Goods-Dominant

Logic (GDL) model [Vargo, 04b]. The existence of other kinds of resources that cannot be distinguished of those that have been identified in GDL is seen as a problematic for services. It has been suggested to stop searching for properties of services that uniquely define them, and instead to view and investigate services as perspectives on the use and offering of resources [Edvardsson, 05]. This model too has its origin of adaptation and extension in the REA model and can be categorized as a generalized conceptual model for the service concept.

2.5 Web Service Description Language

Much of scientific research in service systems are being dominated by web service modeling and conceptualization structural and behavior dependencies of web services [Fensel, 02]. These web service modeling initiatives, e.g., [Fensel, 02] are seeking full-fledged modeling languages for providing the appropriate conceptual model for developing and describing web services and their composition [W3C, 09], [OASIS, 06], and [Preist, 04], and [Lusch, 08]. The web service domain concentrate on Service-Orientated Architectures (SOAs), software systems decomposed into independent units, named as services that interact with one another through message exchanges. The main goal is to promote reuse and evolve-ability, as they start at early phases as possible describing these interactions in the development life-cycle. From standards such as BPEL [OASIS, 06] and WS-CDL [W3C, 09] to languages with purposes derived from their requirements [Schewe, 07] overshadow the service systems and service concept in the contrary to our motivation of a conceptual model for IT service system.

In the reflections of SOA, OASIS it is evident that services are a combination of technical as well as a social concept [OASIS, 06] and that most of the desired expectations in the use of SOA-based systems are rooted in social rather than of physical ones. It is also paramount the creation of value in the context of Service-Dominant Logic in contrast to Goods-Dominant Logic [Lusch, 08].

3 Service Modelling Notations

If a The ambiguity and the overly extensively use of the term service in research and industry has resulted in service oriented technologies, infrastructures and approaches targeting different problems and application domains leading to little or no consensus on its definition, or in the notations used. Often similar ideas with related concepts are developed into service-oriented approaches such as web-services [Platt, 01], and [Walsh, 02] and service oriented architectures, feature-oriented systems in telecommunications [Zave, 01], and [Zave, 03], and service in the middleware technologies [Orfail, 97], and [SUN, 06]. At the light of this confusing use of the term service we look at the service modeling notations from the modeling perspective and service definition contexts used and useful for design and identification of IT services.

3.1 Modelling Services as Components within the Software Development Process

Apparently, service is considered throughout the requirements development process as software components [Meisinger, 08]. According to this view it derives services in terms of software component services for requirements engineering, modeling and architectural design, implementation and integration, and deployment and runtime.

Requirements engineering is concerned with capturing and agreeing on the functional and non-functional properties of systems in a structured way. Classical ways to conduct requirements elicitation and analysis is via structured natural language text using text-based tools such as Word, Doors etc. Usage scenarios are captured as use cases and dependencies depicted as use case diagrams. A multifunctional system is represented as a system offering a number of separate, partially mutually dependent service functions similar to capturing different use cases of a system. Services are pieces of functionality that each providing a partial view on the functionality of the system under certain aspects of usage. In entirety, all service dependencies and interactions make up the systems functionality. Non-functional requirements are associated with the system or its services. Synonyms for this interpretation of service are system function, feature, use case, scenario etc. services can vary in their granularity or degree of preciseness. In case of a precisely defined notation of service, the functional properties of the system can be captured in terms of services. Services can depend on other services. The type of dependency can be further distinguished. Services interact, control or influence each other and theses dependencies are depicted in service dependency graphs [Rittmann, 04].

3.2 Model-Bases Development

Modeled-based development [Schatz, 02], [Braun, 02], and [Broy, 05] helps to handle the complexity that comes with the development of distributed software. Precisely defined models provide abstractions and notational elements for all stakeholders throughout the development cycle. Models are used to provide multiple consistent views on the system and its architecture on different levels of abstraction, tailored for specific intents. The architecture of a reactive system is an essential design artifact in the development process; it needs to effectively support all services of the system in often heterogeneous, distributed environments. The decomposition of a system into its parts and their interconnections determines the further quality of the system, influencing properties such as performance, robustness, maintainability, flexibility to change, etc. Services are used as a way of structuring both modeling process and the models. In service based approaches to architectures, services are the design entities that derive the architecture development and component identification. Services capture defined pieces of functionality and are associated with elements of the software architecture components and their patterns. The service models functionality is provided by interplay of collaborating architectural entities. To describe interactions in the course of service delivery between independent entities sequence diagrams or MSCs [Kruger, 06] are used. The center of concern in model-based design has so far been individual components rather than their inter play. Here the service-oriented development approach can be characterized as a seamless extension of component-based development towards a higher level of abstraction and functional

view on the system with system-wide, component crosscutting functionality. The concept of service decouples abstract behavior from the implementation architectures, by emphasizing the interactions among components.

In *the Implementation Phase* of the development process, the design model is realized by actual hardware and program codes. The implementation varies depending on the applied strategy. In case of model-based development with comprehensive and precise design models, code generation can be performed (semi-) automatically. Manual coding where applied needs to follow the design model to fulfill the designed properties. Implementing a distributed system based on a design model and component architecture is a difficult task when integration of components requires much care and effort to create consistent, efficient and homogenous systems. Important concepts for this regard are packaging and robustness of components, availability of suitable syntactic and semantic component interfaces, powerful tools and flexible infrastructures as the basis of implementation.

Existing service technologies enable to define services as functional interfaces of components or objects. Functionality is encapsulated behind such service interfaces on a suitable level of abstraction and granularity. Such service implementations are usually stateless; they do not depend on the caller or environment state besides the parameters given during the service invocation. This allows a higher degree of deployment flexibility, robustness and decoupling. Service-oriented middleware can provide local or remote access to services and applications are structured as an interplay or workflow of multiple service invocations.

Web service technology defines standards, protocols and methodology in order to provide well defined functionalities as web services, to describe the interfaces using WSDL [W3C, 01], and to connect services using a service-oriented middleware, such as DotNet [Platt, 01] to form device-oriented distributed applications.

Feature-driven development makes features the center of concern. Systems are specified and developed in terms of independent features that are later integrated to form a complete system. The features are based on a system core, providing the basic functionality. Features depend on and overlay each other. There needs to be a defined process to define the dependencies and interactions of the features in order to maintain the consistency and coherence of the system. In such systems, as they occur in the telecommunication domain, extensions happen by adding new features that modify the existing feature functionality or that add new functionality to the system. Features or services are considered as the basic units of increment and change in the systems development process and implementation.

During Service Deployment and Runtime, in order for a system to be executed, the executable code which as mostly packaged in components are required to be distributed across computational nodes, processes, electronic control units, etc. The process of deployment is influenced by infrastructure constraints, hardware and networking layout, reliability, and performance considerations. At this stage the actual efficiency of the system in its environment is determined. The final wirings and configurations of the components are made and adapted to the chosen middleware and network infrastructure; furthermore the component execution life cycle is determined. How and when are components initialized? How many instances and replicas of components are present? How components modified, reconfigured and shut down? During the execution and runtime of distributed systems, services can be used as

distinguishable, modular, executable, deployable entities that make up a service-oriented architecture. Service can be instantiated, initialized, registered and published, looked up, accessed, deployed and dynamically connected. It is important for the flexible service-oriented architectures to have the capability to dynamically look up services during run-time. A central registry or search site needs to be common knowledge in such a system landscape. Services that match certain syntactical criteria and fulfill certain behavioral assumptions can be selected and used immediately. This collaboration is often called the service triangle based on the three riles involved: service provider, service requestor and service registry. Web service infrastructures, for instance, often contain requestor entities, provider entities and registries that communicate using standard protocols such as SOAP for message exchange, WSDL for service description and UDDI for service registration and look up.

In mobile environments, services are the pieces of functionality that networks and mobile applications make available to the user. Availability of certain services is subject to a certain context or location and the service itself might require adaptation depending on the current virtual or physical environment. The environment itself might change, as might the location of the user relative to the environment. Systems need to reflect that, and adapt, reconfigure, and recalibrate.

3.3 Service-Orientation in Application Domains

In web service architectures [W3C 04] the web service systems are built upon following entities: First, service providers, which are technical systems for example agents or servers that make available and perform services for the environment on behalf of humans or business entities. The provided services have a defined description and usage interface. Second, service requesters or consumers and users, select and use the services that satisfy their business needs. Service requesters similarly are technical systems acting on behalf of human or business entities. Third, optionally, a mediating middle instance such as registry or directory, which connects service requests and service providers; it provides service registration and lookup functions to both providers and requesters. Service providers themselves can act as service requesters and thus form higher level services by aggregating or composing other services. An interaction between service requesters and service provider is often called a conversation.

Roots of service-orientation lie in the area of telecommunication. One of the prominent approaches of this domain is very domain-specific in its nature and mainly focus on modeling routing problems and feature interactions [Zave, 03], and [Zave, 00]. For telecommunication systems it is assumed that a base functionality already exists. The main goal of a feature-oriented approach is how to add, remove, modify, and combine pieces of functionality later in the life cycle of such systems. A feature of a software system is an optional or incremental unit of functionality [Zave, 00]. The feature specification contains an action, enabling condition, and priority. The action is performed if the enabling condition is true and the feature has the highest priority. A feature-oriented description is a description of a software system organized by features, consisting of a base description and feature modules, each of which describes a separate feature. The set of possible system behaviors is obtained by applying a composition operator to the base description and the feature descriptions. The composition operator must ensure that in any situation the feature with the

highest priority must be performed. In case of features with a lower priority, their enabling conditions must be changed accordingly.

3.4 Seven Contexts for Service Design

The characteristic concerns and methods of those seven different design contexts is represented in Figure 2 as an unifying view spanning over the information-intensive service systems design and modeling paradigm [Glushko, 10].

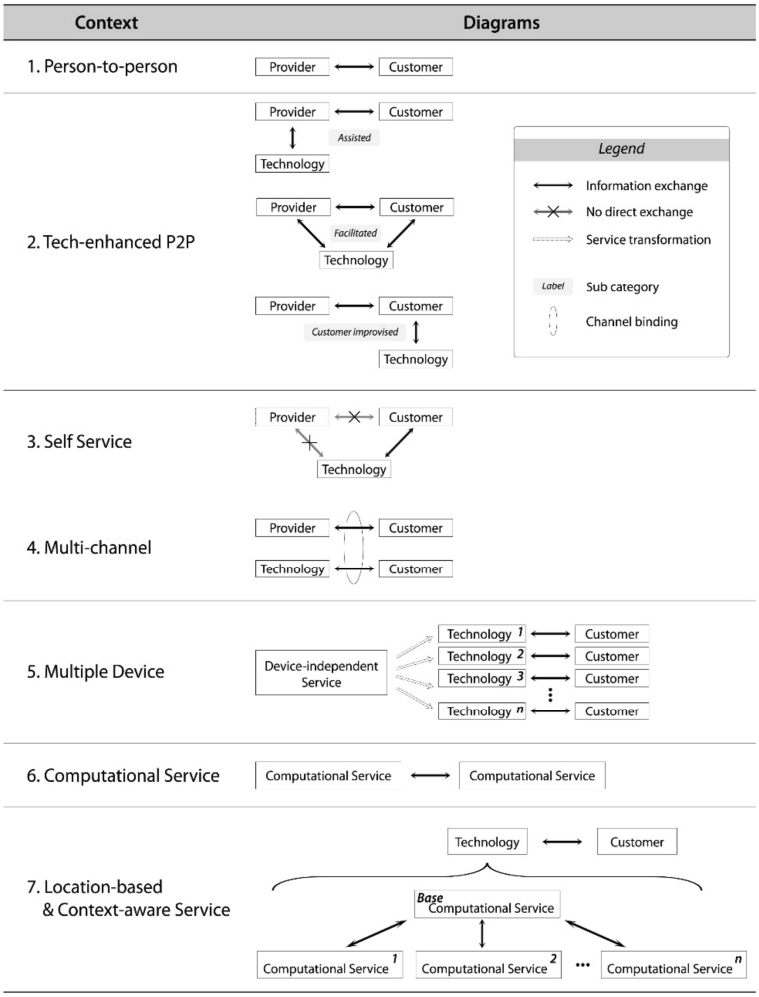

Figure 2: Service modeling notations and contextual representations adopted from [Glushko, 10].

The focus is on the information required to perform the service, how the responsibility to provide this information is divided between the service provider and service consumer, and the patterns that govern information exchange yielding a more abstract description of service encounters and outcomes. Thereby, it makes it easier to see the systematic relationships among the contexts that can be exploited as design parameters or patterns, such as the substitutability of stored or contextual information for person-to-person interactions.

This view of seven contexts for service design [Glushko, 10] reveals the intrinsic design challenges that are inherent within the nature of the information required to perform a service, and emphasizes the need of design choices that allocate the responsibility to provide this information between the service provider and service consumer. The information requirements and the division of labor for satisfying them determine the nature and intensity of the interactions in the service system.

In order to overcome many of the limitations of design approaches that focus more narrowly on the distinctive concerns of each context, a service description language with a more abstract approach that applies to all contexts is required. Therefore, the next sections of this paper will focus on the definition of such a service description language.

A service concept for the service design research has been identified in [Goldstein, 02], [IBM, 04], [Spohrer, 07a] and [Chesbrough, 06] as the key concept for service innovation, design and development and describes a service concept as a how and what of service design and uses as a mediate between customer needs and organizations strategic intents. While the service concept is widely used there is very little has been said in terms of a service innovation, design and development as an IT service system. An IT service system needs to integrate the social, physical and technological aspects in order to provide a service that creates value with social effects. Therefore, in the following sections a general notion for a conceptual model for IT service systems outlined and defined as a framework that separates concerns such as service as a product, service as an offer, service request, service delivery, service application, service record, service log or archive and also service exception, which allows supports a general characterization of services by their ends, their stakeholders, their application domain, their purpose and their context.

4 A Framework and a Conceptual Model for Service Specification

4.1 Service Specification Frames bases on the Framework of Hermagoras of Temnos

The Zachman framework uses the classical W6H description (who – when – where – what – how- why). The key questions in systems development are: ***Who*** will be using the system? ***When*** the system will be used? ***Where*** is the information system used? ***What*** is represented in the system? ***How*** will the system be used? ***Why*** is the system used?

We observe that there are additional dimensions that are of importance: ***Competency***; ***time*** (schedule; delay); ***environment*** (context; technical and organizational); ***quality*** (in which quality; with which guarantees); ***runtime***

characteristics (adaptation; exceptions; delay); ***collaboration*** (with whom, which exchange, on which basis, which portfolio and profile); additional ***motivation*** (on which reason). Additionally, we should take into consideration the policy, intention, goal, and aim of the ***provider***?

One might ask now whether this list is exhaustive and substantial. We also need a prioritization of these questions. Therefore we need an approach that considers the main characteristics in a systematic and surveyable way.

In the classical approach, the specification framework could be headed by the w-questions *who, what, when, where, why, in what way, by what means.* This framework has not been developed in the computer age. It is far older. It dates back to Cicero and even to Hermagoras of Temnos[1] who was one of the inventors of rhetoric frames in the 2nd century BC. The later has been using a frame consisting of the seven questions: Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W7: Who, what, when, where, why, in what way, by what means).

4.2 The General Characteristics of IT Services

Services are to be characterized by their specific properties, the supplier or manufacturer, the pricing that is applicable, and the costs depending on the user, provider and deliverer. Further, services can be kept in an inventory of their providers, their suppliers or their deliverers. Service may be composed of other services. Some information on products is independent of the supplier or provider. Other information, e.g., pricing and availability depends on the supplier.

This approach is used by the Service Modelling Language by W3C. It becomes very sophisticated with many characteristics that must be given. There is no hierarchy in the specification. We use however for characterization two principles: (1) *separation of concern* by aspects and (2) *layering of specifications* depending on the maturity stage. The first principle is the basis for our specification frame that allows us to describe a service. The second principle is used for the specification of an IT service system.

The service is *primarily* declared by specifying

- the **ends** or purpose (*wherefore*) of the service and thus the benefit a potential user may obtain when using the service,
- the **sources** (*whereof*) of the services with the a general description of the environment for the service,
- the **supporting means** (*wherewit*h) which must be known to potential users in the case of utilizing the service, and
- the **surplus value** (*worthiness*) a service utilization might give to the user.

The ***purpose description*** governs the service. It allows one to characterize the service. This characterization is based on the answers for the following questions: *why, whereto, when, for which reason*. We call these properties primary since they define in which cases a service has a usage, usability and usefulness. They define the potential and the capability of the service.

---

[1] The work of Hermagoras of Temnos is almost lost. He had a great influence on orality due to his proposals. For instance, Cicero has intensively discussed his proposals and made them thus available.

A full description of IT services is based on the *four-dimensional specification*:

(a) The ***primary service description*** describes the service.

(b) The ***party dimension*** describes the stakeholders involved into a service. Parties may play different roles, may have different parts in the story of service application, may have obligations, permissions and may also be restricted in their capabilities and competencies. Typical descriptions for the party dimensions are given while answering the *by whom, to whom,* and *whichever* questions.

(c) The ***activity dimension*** describes the processes played during service application. These processes may use resources, may be supported by functions provided by the service system and given at the side of the business user of a service, may result in a number of changes to data, to control and to rights.

(d) The ***application domain dimension*** describes the problems to be solved by the service, the application area in which a service is usable, the typical approaches within this application area, the typical solutions that are sought for the problems under consideration. We thus answer questions like *wherein, where, for what, wherefrom, whence, what*.

Additionally we might also declare the context characteristics for a service. Context has at least four sides: *provider* or developer or supplier *context* for a service, the *user context* for a service, *the system environment context* that must exist for service utilization, and the *coexistence context* for a service within a set of services. Therefore, these context dimensions are declared by answering the following questions: *whereat, whereabout, whither, when*.

To summarize, our service description language is thus based on the following questions:

- Primarily: ***wherefore, whereof, wherewith, worthiness,*** and additionally ***why, whereto, when, for which reason****;*
- Secondarily*:* ***by whom, to whom, whichever; wherein, where, for what, wherefrom, whence, what; how***;
- Additionally: ***whereat, whereabout, whither, when***.

We may call our framework the (***W4 + W4 + W14H***) specification frame. The kernel of this framework is the (W4 + W10H) questionnaire.

4.3 The Parties Involved in Services

Parties in services have their own target, goal, intentions, and aims. They are bound by their capabilities, support requirements, and information demand. This description constitutes the kernel of the *profile*. Other components of the profile of a stakeholder in a service process include educational, employment, psychological descriptions.

Parties are interested in solving a certain number of tasks. Tasks are ordered and prioritized. A task is an assigned piece of work, which often has to be finished within a certain time by a party or parties whose duty is its completion. It implies work

imposed by a user in authority and obligation or responsibility to perform. A task may consist of subtasks, so we assume that tasks can be constructed on the basis of elementary tasks.

The *portfolio* consists of an ordered and prioritized collection of tasks including compensation and is determined by the responsibilities one has and is based on a number of targets. The party portfolio within an application is thus based on a set of tasks a actor has or intents to complete and for which solution the actor has the authority and control, a description of involvement within the task solution, a collaboration that is necessary for solving the task. Tasks are supported by services. A party may activated a service in order to complete a task.

The *involvement* of parties within some service activation is based on the specification of the *role* a party plays during execution of the service, the *part* the party plays within its portfolio, and the *rights* and *obligations* a party has within the given role. The role specifies the behavior expected for a party. A role is a comprehensive pattern of behavior and serves as a strategy for coping with recurrent situations and dealing with the roles of others. A role remains relatively stable, even though different parties occupy the position. A party may have a unique style of role execution, but this is exhibited within the boundaries of the expected behavior of the party. Role expectations include both actions and qualities. There are two types of roles: declarative and contextual ones. The former ones declare that a party is playing a particular role, e,g., a party being identified and as an employee. Contextual roles show how a party acts within the context of an application story and how it is involved within the context of another application story. Declarative roles may be modeled by associating the actor to a role type. Contextual roles are modeled by associating an actor with the work effort the actor is assigned to and a role type describing the involvement of the actor. The role type provides a description of the role and can be hierarchically structured. Roles may also be hierarchically structured. At the same time roles may be played in collaboration.

4.4 The Service Activities

Services are typically based on a (set of) workflow(s) that must be followed by party. These services are often virtual goods. A service is *offered* together with policies applicable to the service export. A *service provider* can advertise, modify and withdraw a service. Service providers are often supported by *traders* that store services available from service providers and act as brokers for available services and actual requests. Traders are able to search for the most appropriate service for a given request on the basis of matching criteria and search constraints. They store new advertisements of service providers and categorize them. If the trader has a large variety of services available from providers then the service offer properties are standardized to service offer property types. A trader has further a trading offer domain and is restricted by trading contexts. The information schema of the traded services enables is maintenance of meta-structures for services, categorization and partitioning. Traders themselves may be organized in trading syndicates or trading communities with trading administrations, internal arbitraries, makers for trading rules and policies, export policy control, and trader owners. Trader communities can be federated with external arbitrators.

Services are typically layered depending on the stage a service is used by a party. We distinguish the following stages: First a service is developed. This stage is similar to a production. Next a service is quoted. A party may now use the service and request it. The supporting party responds and provides details on the service use. Next is requesting and ordering of the service. Later the service is delivered. Finally, the service must be billed and paid. Parties involved into this service utilization process play different roles and parts and have different responsibilities. This service utilization process is typically layered as displayed in the following Figure 3. It can also be chained or executed partially in parallel or in different chain.

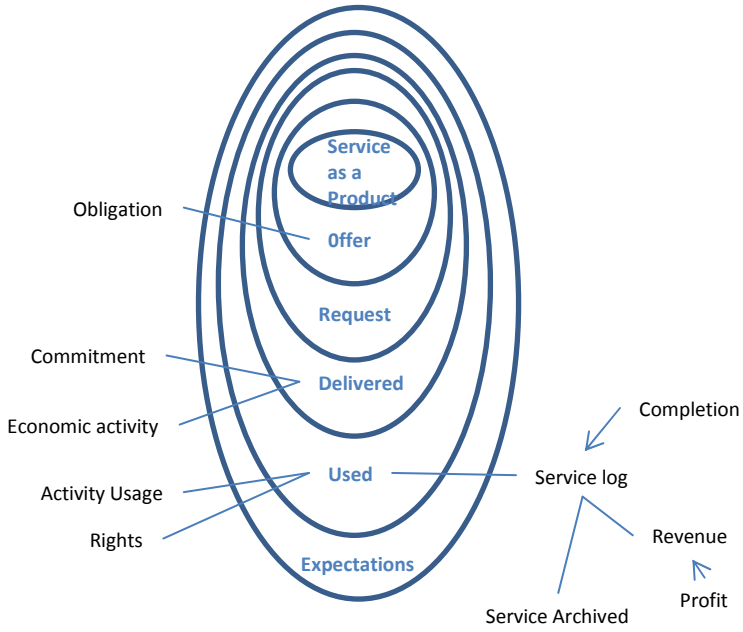

Figure 3: The layered services utilization process

4.5 An Example of our Service Description Framework

We demonstrate our approach using a special service system that supports ventilation therapy. Ventilated patients are among the most vulnerable patients in the hospital. A typical example of such system is the Evila XL from Draeger (http://www.draeger.com/US/ en_US/applications/ventilation_therapy/). The entire ventilation support process must however be reduced. Otherwise due to the increased risk for infections, it results in ventilator-induced injuries and higher mortality rates after failed extubation. Evila XL from Draeger (http://www.draeger.com/US/ en_US/applications/ventilation_therapy/) focuses on the entire ventilation process and improves it for patients. It addresses thus the following challenges: Avoid intubation when possible; easy tailor treatment; encourage spontaneous breathing; practical and sophisticated transport systems; wean efficiently and reduce reintubations; avoid complications and support recovery. To optimize therapy for patients of any age and any acuity level, Dräger provides ventilators designed to support efforts to improve

patient outcomes and reduce complications. This system is a real life service system and may thus serve as an example of service description.

The Service Description for Evita XL from Draeger

Service

(1) **End** (wherefore)
- *Partially supported regulation of deep breading*

(2) **Sources** (whereof)
- *Computerized ventilation equipment*

(3) **Supporting means** (wherewith)
- *Monitoring and relating devices*

(4) **Surplus value** (worthiness)
- *Releases nurses from routine observations and maintenances*

(5) **Purpose** (why, whereto, when, for-which-reason)
- *Insufficiency or unconscious patient*
- *Breathing system*
- *Observed insufficiency*
- *Support for regulating breathing*

(6) Activity

Input (what-in)
- *Ventilation data*

Output (what-out)
- *Ventilation controller*

(7) Party

Suppliers (by-whom)
- *Emergency ward staff*

Consumer (to-whom)
- *patient*

Producer (whichever)
- *IMV company*

(8) Application domain

Application area (wherein)
- *Breath intermittent mandatory ventilation (IMV)*

Application Case (wherefrom)
- *After surgery or accident care*

Problem (for-what)
- *Breath emergency*

Organizational unit (where)
- *Emergency unit*

Triggering events (whence)
- *Breath insufficiency of patient*

IT {data, control, computation} (what, how)
- *Body monitor*
- *data*

(9) Context

System context (whereat)
- *Emergency chamber environment*

Story context (where-about)
- *Breath insufficiency of patient*

Coexistence context (whither)
- *Integratable masks*

Time context (when)
- *Scheduled at maximum for one week*

There are several common capabilities and characteristics across the entire ventilation product line: **Complete solutions** target at reduced risk of infection by including disposable supplies and accessories that enable ventilation without sterilization. **Ease-of -use and intuitive design** simplifies operation and contributes thus to patient safety while enhancing user proficiency and overall productivity. The **Open breathing system with Room to Breathe™** allows spontaneous breathing during ventilation.

Non-invasive ventilation (NIV) capabilities are available in most Dräger ventilators. NIV in general has been shown to reduce the need for intubation.

5 Conceptual Model for IT Service Systems

Our main objective is to We apply layering for the conceptual model of an IT service system. For any system we may distinguish the supply profile and the demand portfolio. Applications have their specific demands. An IT service system provides a number of services. These services are combined within a supply profile of the IT service system.

5.1 The Service Profile

A profile specifies the services and the capability of a system. For any service we may distinguish the supply profile and the demand portfolio. Applications have their specific demands. A service system may provide a number of services. These services are combined within a supply profile. If there is a mismatch then it might result in a performance problem or in insufficient utilization of resources. A service profile declares the services within the frame introduced above and describes the capability of a system.

5.2 The Service Portfolio

A service portfolio consists of a set or collection of tasks that a service user might solve by the given service. A *portfolio* consists of a set or collection of tasks that can be solved by utilization of the service. A portfolio is determined by the responsibilities one has and is based on a number of targets one has.

The *party portfolio* within an application is thus based on

- a set of *tasks* a party has or intents to complete and for which solution the party has the authority and control over,

- a description of involvement within the task solutions, and
- a collaboration that is formed for the tasks solution.

Task modeling means to understand what a user wants to accomplish while utilizing the IT service system. At the same time, task analysis may lead to a reorganization of the work processes to be supported. Task analysis leads to a description of things users do, of things they act on, and of things they need to know. It does not specify how the task is accomplished. The tasks need to be representative for the application, important within the application, and completely supported. Task support can be tailored depending on the profile and the context of the parties.

Service utilization is formed according to tasks to be solved. Each of the parties has a portfolio that consists of all tasks and that defines the involvement, collaboration and restrictions.

The *party portfolio* uses a party portfolio name. Task specification includes a general description and characterization of the task, a characterization of initial states, and a characterization of target states. Additionally it may include a description of a generic profile presupposed for solution, a list of instruments used for solution, style/pattern for service utilization, and a list of auxiliary conditions.

Execution description is based on a list of activities, control, and data. The targeted result is described through a final state and target conditions

Party involvement is based on a general description of the intentions of the party while using a service, on a description of roles the party plays, on a description of the part of the party in the play based on behavioral categories/stereotypes.

A service portfolio may also use a set of restrictions for their utilization.

5.3 Many-Layered Architecture of Services

This separation into profile for declaration of capabilities and portfolio for declaration of requests allows us to specify requests issued by an application through the application demand portfolio and to describe the services provided by the service provider or trader through the service supply profile. In a similar form we describe the demand of a service system by the service demand portfolio and the services provided by the support systems and by the system supply profile. The service supply profile consists of the characterization of the service system, the extended data dictionary or namespace that is used for the service system, and a specification of system's variables and parameters, and the utilities.

6 Discussion and Future Research Work

Most complex IT service systems that are being built or envisioned nowadays combine person-to-person encounters, technology-enhanced encounters, self-service, computational services, multi-channel, multi-device, and location based and context-aware services [Glushko, 10]. This paper bases the service description on distinction through characteristic concerns. It thus provides a unifying view of all design contexts, especially in the case of service systems that are "information-intensive" IT service system. The full description of IT services is based on the *five-dimensional specification*: (1) primary service description (end, sources, supporting means, surplus

value, purpose), (2) party dimension, (3) activity dimension, (4) application dimension, (5) context dimension. We may structure this description into primary, secondary and additional questions.

A focus on the (***W4 + W4 + W14H***) specification yields to a more abstract description of service. The description of a service combines viewpoints of the service provider and of the service consumer. The *service deployment* is layered into (a) the service description, (b) the service offer, (c) the service request, (d) the service delivery, (e) the service use, and (f) the expectation of the user that deploy the service. We may use general stories of service deployment. Those might be governed by patterns or styles of information exchange.

Future research explores, tests and evaluates our IT service systems specification language for innovation, design and development of IT service systems in the seven context classification of person-to-person encounters, technology-enhanced encounters, self-service, computational services, multi-channel, multi-device, and location based and context-aware services. Future research is going to investigate the application of the IT service systems specification language to other real-life cases. [Glushko, 10] claims that the systematic relationships among the contexts can be exploited as design parameters or patterns. Our abstract specification language on IT service systems design could turn different design context into building blocks that enable the incremental design of service systems. Most IT service designers are familiar with some of these contexts, and each context has a research and practitioner literature that highlights their characteristic design concerns and methods. But few service designers are familiar with all of them. The design concerns and methods in one context seem to be incompatible with those in others. There is relatively little work that analyses design concerns and methods which span multiple contexts [Glushko, 10].

In addition, unifying ideas about service interfaces and information exchange extend the description of service interfaces and of the information exchange. In this case we may understand the impact of and problems for intersection of IT service systems. We need to explore composition and interaction of services.

References

[Arsanjani, 08] Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: A method for developing service-oriented solutions, IBM Systems Journal 47(3): 377-396, 2008.

[Bergholtz, 10] Bergholtz, M., Andersson, B., Johannesson, P.: Abstraction, Restriction, and Co-creation: Three Perspectives on Services, ER 2010 Workshops of Conceptual Modeling of Services, LNCS 6413, 107-116, 2010.

[Bosworth, 03] Bosworth, B.P., Triplett, J.E.: Services productivity in the United Sates: Griliches' services volume revisited, The Brookings Institute, Washington, DC, 2003.

[Braun, 02] Braun, P., Beeck, v-d, M., Rappl, M., Schroder, C.: Automotive Software Development: A Model-Based Approach, In-vehicle Software, SEA Technical Series, 2002.

[Broy, 05] Broy, M.: Service-Oriented Engineering: Specification and Design of Services and Layered Architectures – The Janus Approach, In: Engineering Theories of Software Intensive Systems, 47-81, Springer, 2005.

[Bryson, 04] Bryson, J.R., Daniels, P.W., Warf, B.: Service World: People, technology, Organizations, Routledge, London, 2004.

[Chen, 09] Chen, N., Dahanayake, A.: Role-based situation-aware information seeking and retrieval service design approach for crisis response, International Journal of Information Systems for Crisis Response and Management (IJSCRM), 11(3), 2009.

[Chesbrough, 06] Chesbrough, H., Spohrer, J.: A Research Manifesto for Service Science, CACM 49, 35-40, 2006.

[Spohrer, 07a] Spohrer, J., Maglio, P.P., Bailey, J., Gruhl, D.: Steps Towards a Science of Service Systems, IEEE Computer 40, 71-77, 2007.

[Edvardsson, 05] Edvardsson, B., Gustafsson, A., Roos, I.: Service portraits in a service research: a critical review, Int. J. of Service Industry Management 16(1), 107-121, 2005.

[Erl, 07] Erl, T.: SOA: principles of service design, Prentice-Hall, Englewood Cliffs, 2007.

[Fensel, 02] Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF, Electronic Commerce: Research and Applications, 1(2), 113-137, 2002.

[Geerts, 99] Geerts, G., McCarthy, W.E.: An Accounting Object Infrastructure for Knowledge-Based Enterprise Models, IEEE Int. C. Systems & Their Applications. 89-94, 1999.

[Glushko, 10] Glushko, R.J.: Seven Contexts for Service System Design, In P.P. Maglio et al. (eds.), Handbook of Service Science, Service Science: Research and Innovations in the Service Economy, DOI 10.1007/978-1-4419-1628-0_11, Springer Science+Business Media, LLC 2010.

[Goldstein, 02] Goldstein, S.M., Johnston, R., Duffy, J-A., Rao, J.: The service concept: the missing link in service design research?, Journal of Operations Management 20, 212-134, Elsevier, 2002.

[Hirschheim, 10] Hirschheim, R., Welke, R.J. , and Schwarz, A.: Service Oriented Architecture: Myths, Realities, and a Maturity Model, MIS Quarterly Executive 9(1), 204-214, 2010.

[Hurby, 06] Hurby, P.: Model-Driven Design of Software Applications with Business Patterns, Springer, Heidelberg, ISBN: 3540301542, 2006.

[IBM, 04] IBM Research.: Service Science: A New Academic Discipline?, available at http://www.almaden.ibm.com/asr/resources/facsummit.pdf, 2004.

[Kruger, 06] Kruger, I., Mathew, R., Meisinger, M.: Efficient Exploration of Service-Oriented Architectures Using Aspects, In Proceedings of the 28th International Conference on Software Engineering (ICSE), 2006.

[Lusch, 08] Lusch, R.F., Vargo, S.L., Wessels, G.: Toward a conceptual foundation for service science: contributions from service-dominant logic, IBM Systems Journal, 47(1), 2008.

[Maglio, 06] Maglio, P., Srinivasan, S., Kreulen, J., Spohrer, J.: Service Systems, Service Scientists, SSME, and Innovation, Communications of the ACM, 49(7): 81-85, 2006.

[McCarthy, 82] McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment, Accounting Review 57, 554-578, 1982.

[Meisinger, 08] Meisinger, M., Rittmann, S.: A Comparison of Service-oriented Development Approaches, Technical paper (TUMI-0825), Technical University of Munich, Germany, 2008.

[OASIS, 06] OASIS.: Reference Model for Service Oriented Architecture 1.0, http://www.oasis-open.org/committees/download.php/19679/, 2006.

[Orfail, 97] Orfail, R., Harkey, D., Edwards, J.: Instant CORBA, Wiley, 1997,

[Papazoglou, 06] Papazoglou, M. P.,van den Heuvel, W-J.: Service-oriented design and development methodology, Int. J. Web Eng. Technol. 2(4): 412-442, 2006.

[Platt, 01] Platt, D.S., Ballinger, K.: Introducing Microsoft .NET, Microsoft Press, ISBN: 0-7356-1377X, 2001.

[Preist, 04] Preist, C.: A Conceptual Architecture for Semantic Web Services, In: Mcllraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWS 2004, LNCS-3298, 395-409, 2004

[Poels, 10] Poels, G.: The Resource-Service-System Model for Service Science, ER2010 Workshops, LNCS-6413, 117-126, 2010.

[Rittmann, 04] Rittmann, S.: Exploring Service-Oriented Software Development for Automotive Systems, Masters' Thesis, Technical University of Munich, Germany, 2004.

[Schatz, 02] Schatz, B., Pretschner, A. Huber, F., Philipps, J.: Model-Based Development of Embedded Systems, Technical Report (TUMI-0402), TU Munich, Germany, 2002.

[Schewe, 07] Schewe, K-D., Thalheim, B. (2007): Development of Collaboration Frameworks for Distributed Web Information Systems, Proc. IJCAI'07 (20th Int. Joint Conf on Artificial Intelligence, Section EMC), 27-32, 2007.

[Stojanovic, 04] Stojanovic, Z., Dahanayake, A.: Service – Oriented Software Systems Engineering: Challenges and Practices, Idea Group Publishing, PA, USA, 2004.

[Spohrer, 07b] Spohrer, J., Maglio, P., Bailey, J., Gruhl, D.: Steps Towards a Science of Service Systems, IEEE Computer, 40(1): 71-77, 2007

[Spohrer, 08] Spohrer, J., Maglio, P.P.:The Emergence of Service Science: Towards systematic service innovations to accelerate co-creation of value, IBM Almaden Research Center, 2008.

[SUN, 06] SUN Microsystems Inc.:Java Platform, Enterprise Edition (Java EE, J2EE), Available at http://java.sun.com/javaee/, Version of Nov-15-2006

[Vargo, 04a] Vargo, S.L., Lusch, R.F.: Evolving to a New Dominant Logic for Marketing, Journal of Marketing 68, 1-17, 2004.

[Vargo, 04b] Vargo, S.L., Lusch, R.F.: The Four Services Marketing Myths: Remnants from manufacturing model, Journal of Service Research 6, 324-335, 2004.

[Vargo, 08] Vargo, S.L., Maglio, P.P., Akaka, M.A.: On value and value co-creation: A service systems and service logic perspective, European Management Journal 26, 145-152, 2008.

[W3C, 01] W3C: Web Description Language 1.1, Available at: http://www.w3.org/TR/wsdl. 12-Mar-2001.

[W3C 04] W3C: Web Service Architecture, Specifications available at: http://www.w3.org/TR/ws-arch/, 11-Feb-2004.

[W3C, 09] W3C Working Group.: Web Service Modeling Language, Version 1.1, http://www.w3.org/TR/sml/, W3C Recommendation 12 May 2009.

[Walsh, 02] Walsh, A.: UDDI, SOAP, and WSDL: The Web Service Specification Reference book, Prentice Hall, 2002.

[Zave, 00] Zave, P., Jackson, M.: New Feature Interactions in Mobile and Multimedia Telecommunication Services, Feature Interaction in Telecommunications and Software Systems V!, pp 51-66. ISO Press, available at http://www.research.att.com/~pamela/fiw6.pdf, *IEEE Computer*, 40(1): 71-77, 2000.

[Zave, 01] Zave, P.: Feature-Oriented Description, Formal Methods, and DFC, In proceedings of the FIREworks Workshop on Language Constructs for Describing Features, 11-26, Springer-Verlag, 2001.

[Zave, 03] Zave, P.: An Experiment in Feature Engineering, In: Annabelle McIver and Carroll Morgan, editors, Programming Methodology, pp 353-377, Springer-Verlag, New York, 2003.

[Zimmerman, 04] Zimmerman, O., Krogdahl, P., Gee, C.: Elements of service-Oriented Analysis and design". http://www-128.ibm.com/developerworks/library/ws-soadl, 2004.