International Journal of Digital Information and Wireless Communications (IJDIWC) 2(4): 276-295
The Society of Digital Information and Wireless Communications, 2012 (ISSN: 2225X-658X)

A FRAMEWORK FOR MODELING MEDICAL DIAGNOSIS AND DECISION SUPPORT SERVICES

Sankha Amarakoon [1], Ajantha Dahanayake [2], Bernhard Thalheim[3]
[1] Rotterdam Ophthalmologic Institute, Rotterdam, the Netherlands.
S,Amarakoon@oogziekenhuis.nl
[2] Prince Sultan University, Dept. of CIS, Riyadh, Kingdom of Saudi Arabia.
adahanayake@pscw.psu.edu.sa
[3] Christian-Albrechts-University, Dept. of Computer Science, 24098 Kiel, Germany.
thalheim@is.informatik.uni-kiel.de

ABSTRACT
This paper introduces a framework and a service model improving the understanding of domain requirements acquisition for IT-service systems development. The service model fills the gap in domain specific requirements elicitation through its base in the classical rhetorical frame introduced by Hermagoras of Temnos and through the interpretation of the domain in terms of service offerings. The model is validated by a real-life situation – to establish an ophthalmologic Disease Diagnosis Decision Support Network (DDDSN) for age-related macular degeneration (ARMD) treatments.

KEYWORDS
Requirements Modeling, Service Systems (IT), Cross-Disciplinary Application Development, Specification Frame, Age Related Macular Degeneration, Medical Diagnosis Decision Support, Service Model

1 INTRODUCTION

The field of Information Systems (IS) is shifting towards advanced and cross-disciplinary IT-service systems engineering. For example, Medical, Environmental, Disaster Recovery, Life-Science, etc., are such domains that largely invest on advanced cross-disciplinary IT-service systems. The implication of this shift is that those IT-service systems are then subject to evaluations of systems functioning based on its trustworthiness, flexibility to change, and efficient manageability and maintainability. Those systems development endeavors demand their developers to understand the systems functioning within its domain of usage and its service. Today, organizations opt to define, develop and deploy cross-disciplinary IT-service systems, making those resulting applications available for end users to amalgamate or mash-up those into end-user-situational services in ways that the developers may not originally envision [1].

1.1 IT Service Systems

IT service systems combine and integrate the value created in different design contexts of; person-to-person encounters, technology enabled self-service, computational services, multi-channel, multi-device, and location-based and context-aware services [2]. There is a substantial subset of IT service systems that can be described as “information-intensive”, and it takes a more abstract view of service contexts that highlights what person-to-person, self-service, and automated or computational services are.

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 [3]. The missing link in IT-service systems design and development is the service concept itself [4]. A service centered domain requirements elicitation can contribute to the demarcation of the domain in terms of services as well as the use of terminology specific to the domain leading to an improved understanding of the domain [4] [5].

IT-service systems are typical software systems that concentrate their operating on data or information they provide. None the less an IT-service system must comply with its functioning according to the needs and rules of the domain it serves. IT-service systems must serve a domain consisting of a comprehensive set of service offerings. The understanding of domain and its requirements are more important than ever as we extend ourselves to cross-disciplinary IT-service systems development.

From solution modeling to domain requirements modeling

Domain engineering focuses on capturing knowledge gathered during the software engineering process. Domain Engineering stems from software product line designing to improve the quality of developed software products through reuse of software artifacts [25] and claims that most developed software systems are not new systems but rather variants of other systems within the same field [25]. Classically it concentrates on the development of reusable artifacts dealing with specifying, designing, implementing, and managing reusable assets, such as specification sets, patterns, and components, that may be suitable, after customization, adaptation, or even extension, to families of software product components to be reused in new software systems at low cost and higher quality [26]. This produces not only a set of software implementation components relevant to the domain, but also reusable and configurable requirements and designs. As a result, through the use of domain engineering, businesses can maximize profits and reduce time-to-market by using the concepts and implementations from prior software systems and applying them to the target system [25] [26].

Domain engineering, like application software engineering, consists of three primary phases: analysis, design, and implementation. However, where software engineering focuses on a *single* system, domain engineering focuses on a *family* of systems [26]. Domain analysis is used to define the domain, collect information about the domain, and produce a domain model [26]. Through the use of feature models (initially conceived as part of the feature-oriented domain analysis methods), domain analysis aims to identify the common points in a domain and the varying points in the domain [26]. A domain specific language aims at designing and developing languages that support the terminologies of domains and at defining, implementing, and validating syntactic and semantic rules that enable combinations of terms [27]. A basis theory for application domain languages is currently under development, e.g. based on description logics, artificial intelligence, logical calculi, ontologies, and VDM.

A good domain model could serves as a reference to resolve ambiguities later in the process, with a repository of knowledge about the domain characteristics and definition, and a specification to developers of products which are part of the domain [29]. It is possible through the use of domain analysis, the development of *configurable* requirements and architectures, rather than *static* configurations which would be produced by a traditional application engineering approach. Conceptual modeling contributions of Domain Engineering is on capturing, representing, analyzing and processing knowledge about the solution and how this knowledge can contribute to software engineering, and do not contributed to explore conceptualization in terms of understanding the domain [30].

D. Bjorner [6] divides software engineering into three main phases: application domain description, requirements prescriptions, and systems specifications. He calls these phases the *system development triptych*. The application domain description is a model describing the application domain, its entities, functions, events, and behavior. It is based on a formal, semi-formal or natural language which allows us to formulate a set of theorems or postulates or properties that are claimed to behold of the domain model. The challenge of application domain description and modeling is: to achieve a set of languages, principles, methods, theories and techniques; as well as a set of management practices; which together cover all of today's and the immediately foreseen applications, their variations and their evolution in future; which by careful use and thoughtful consideration support software systems during evolution from initial development via repeated adaptive and perfective maintenance to final disposition, and which ensure a near bug-free software as is humanly conceivable.

The *grand* challenge of application domain modeling is the evolution of the application domain itself. This challenge becomes crucial for modern IT-service systems such as web information systems (WIS), due to the low 'half-life' period observed for these systems applications and the high potential to evolution, migration, and integration. Therefore, the incorporation of a service-offering perspective into application domain requirements modeling approach have a great potential to contribute to advanced cross-disciplinary systems building in the large.

Problems of IS Systems Modeling Techniques

IS modeling techniques suffer from a number of weaknesses [7]. These weaknesses are mainly caused by concentration on database modeling [23], on the functionalism applied and by non-consideration of application domain problems [24] that must be solved by IT-service systems.

Unfortunately, domain models in the field of domain engineering are heavily influenced by this database solution modeling practice of capturing, representing, analyzing and processing knowledge about the solution and how this knowledge can contribute to software engineering. Conceptual modeling has much to gain in contributing to the domain understanding [30]. Therefore, this paper presents a conceptual model and a domain requirements modeling language for engineering cross-disciplinary IT-service systems.

The paper is organized as follows: Section 2 is an overview of a real-life IT service systems development initiative from the domain of medicine and thus highlights the uniqueness of our contribution. Section 3 provides an evaluation of the state of research initiatives that have been contributing to developing a service model. Section 4 summarizes the classical rhetorical frame introduced by Hermagoras of Temnos which influenced our philosophical foundation and conceptual model of the Domain Requirements Modeling for IT-service systems. Section 4 outlines the application of Domain requirements modeling for Ophthalmologic DDDSN and its contribution to grounding of aim and goals for solution modeling. Section 5 describes the different concerns of services depending on the phases. As we need a more detailed view to the notion of a service. We conclude in Section 6 providing conclusions, and future research initiatives.

2 CROSS-DISCIPLINARY IT-SERVICE SYSTEMS

Ophthalmology covers around 3,000 eye diseases and is identified using images of the eye. The images are a vital part of the knowledge and the characteristics of the diseases leading to effective treatment procedures [8]. For each of the diseases there is a collection of images which are predominantly assets of the specialist and are not accessible or used by other specialists. Those collections of images and accompanying knowledge of treatments are a vital part of the knowledge that is available but not harnessed in a useful and effective manner to treat patients irrespective of their country of residence or their life situation.

An Ophthalmologic Research Institute (ORI) in the Netherlands initiated to establish a DDDSN for ophthalmologists. A knowledge repository of images of ophthalmology diseases and building decision support through a location-independent disease diagnosis decision support (DDDS) coordination. The participating ophthalmologists contributing to build the ophthalmology image repository incrementally and use the network to organize focused working groups, and reach out to collaborating groups to establish an expert community of ophthalmologists. The motivation behind this is to harvest tangible benefit in the form of shared access to this unique image repository to achieve interoperability and location-independent decision support, for the benefit of specialists from all over the world, and to be able to treat their patients with up-to-date disease diagnosis decision support.

Medical diagnosis decision support systems

Medical diagnosis decision support is hardly in existence [9]. An impressive amount of medical images are daily generated in hospitals and medical centers. Consequently, the physicians have an increasing number of images to analyze manually [10]. A number of researchers have identified that computational techniques can provide assistance to physicians in analyzing digital images [11] [12]. One of the problems associated with computerized medical diagnosis is the mounting resistance due to the fear of being replaced by a computer and in turn losing the prestigious position of medical practitioners. Therefore, DDDS are hardly available [9] [10] [11].

In the practice, an ophthalmologist's diagnoses are not contested or validated by another specialist. Access to a system that generates a second opinion has an added value in serving as an extra pair of eyes without violating the autonomy, professionalism or credibility of the ophthalmology profession. Such a system contributes vastly to the decision making process in all medical fields [13], [14], and it has greater odds of acceptance by specialists.

Age-related Macular Degeneration (ARMD)

Age-related macular degeneration (ARMD) is the main cause of decreased visual acuity in patients older than 65 years in the Netherlands and the rest of the western countries [15]. ARMD results in a deterioration of the central retinal function and is the leading cause of blindness in people over 65 years of age in Europe and the U.S. Because of the localization of the macula in the center of the retina, advanced age-related macular degeneration often leads to irreversible loss of social skills, such as reading ability. Two forms of ARMD are distinguished: the atrophic form and the neovascular, exudative, or wet form.

The atrophic form, typically, involves the choriocapillaris, retinal pigment epithelium (RPE), and photoreceptor elements (rods and cones) and does not involve leakage of blood or serum; hence, it is called dry ARMD. The neovascular, exudative form includes serous or hemorrhagic detachment of RPE and choroidal neovascularization, which leads to leakage of blood and serum; hence, it is called exudative or wet ARMD. Wet ARMD progresses to fibrovascular scarring of the macular area.

Loss of vision can occur in either of the two forms of the disorder. The wet form, with choroidal neovascularization, is more aggressive and may progress more rapidly to blindness. Among patients with severe loss of visual acuity, choroidal neovascularization (CNV) is the cause in at least 80% of the cases. Leakage of blood or serum as a result of CNV may occur precipitously and is often associated with the abrupt loss or distortion of vision. CNV can be identified before scarring and extensive leakages have caused irreversible loss of vision. Currently, there are no treatment options for the atrophic form of ARMD. In 2007, treatment options became available for the treatment of wet ARMD. These drugs have to be administered by monthly intravitreal injection and as a result have increased the workload for ophthalmologists extensively. This increase is not only a result of the frequent injections but also because images of the retina (fundus photography, optical coherence tomography (OCT) scans) are required for the follow-up of disease progression during the treatment period as well.

The proposed advanced cross-disciplinary DDDSN-system has three services for practicing ophthalmologists: (1) decision support image base, (2) continuous knowledge enhancement of the image base, and (3) an on-demand learning module for specialists and residents to update their diagnosis and knowledge.

DDDSN initiative exposes the systems designing to the challenges of the evolution of the application domain. This is a crucial challenge for modern cross-disciplinary IT-service systems, as per definition they are web information systems, are notorious for their low 'half-life' period and the high potential of evolution, migration and integration. Therefore, it is necessary to incorporate application domain requirements modeling into such systems building approaches.

In the following section we will explore the service concept, its frameworks, and conceptualization approaches in order to formulate a service modeling approach for domain requirements acquisition.

3 EXPLORING THE SERVICE CONCEPT AND ITS FRAMEWORKS

Service science is an initiative of IBM [43] 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 [55]. As a distinct interdisciplinary field, it searches for an ideology and a unifying paradigm [43].

The Resource-Event-Agent (REA) 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 [46] was further articulated and extended by others, e.g., [40] and [42]. 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 [42]. 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.

The RSS Model

The Resource-Service-Systems (RSS) model for service systems [51] 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) [57] 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 [57]. SDL sees all economic activity as service exchange between service systems [57]. In SDL, service is a competence that exchanges for the benefit of other service systems. In contrast to traditional Goods-Dominant Logic (GDL) model [57], 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.

The Three Perspectives on Services

The abstraction, restriction, and co-creation is introduced as a conceptual model of service concept that views services as perspectives on the use and offering of resources [34]. 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 [57]. 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 [38, 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.

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 [39]. These web service modeling initiatives, e.g., [39] are seeking full-fledged modeling languages for providing the appropriate conceptual model for developing and describing web services and their composition [60], [47], and [50], and [45]. 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 [47] and WS-CDL [60] to languages with purposes derived from their requirements [54] 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 [47] 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 [45].

Service Modeling Notations

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 [49], and [61] and service oriented architectures, feature-oriented systems in telecommunications [62], and [62], and service in the middleware technologies [48], and [56]. 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.

Modelling Services as Components within the Software Development Process

Apparently, service is considered throughout the requirements development process as software components [28]. 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 [52].

Model-Bases Development

Modeled-based development [53], [35], and [36] 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 [44] 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 systemwide, 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 [58], and to connect services using a service-oriented middleware, such as DotNet [49] 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.

Service-Orientation in Application Domains

In web service architectures [59] 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 [62], and [62]. 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 [62]. 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.

Seven Contexts for Service Design

The characteristic concerns and methods of those seven different design contexts is represented in an unifying view spanning over the information-intensive service systems design and modeling paradigm [3]. 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 [3] 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 [41], [43], [55] and [37] 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.

In the following section we introduce a novel conceptual model for service systems modeling.

4 THE CONCEPTUAL MODEL FOR SERVICE MODELING

A well-known fact in systems engineering is that we must thoroughly understand the requirements of the intended IT system before we dive into development and it is true for developing cross-disciplinary IT-service systems, thus we must understand its requirements. The requirements elicitation is only successful when we have understood the functioning domain of the potential IT-service system. The term domain engineering has been around for some time in the software systems engineering field. Domain is an area of human activity or an area of science, generally referred to as a universe of discourse. Domains are named with sufficiently well delineated and justifiable names and are well distinguished from neighboring universes or areas to avoid unnecessary overlap and confusion. Some examples of domains are: air traffic, financial service industry such as, banks, insurance companies, portfolio managers, stock brokers, traders and exchanges, etc., [31]

Ideally, domain requirements describe an idealized view of the domain [31] and in a more abstract manner domain can be described by a collection of services offerings. Informally, we define domain requirements modeling as the description of domain's IT services and requirements expressed using the terms of the domain.

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 [4]. The service concept defines the what, how and who on what basis of service innovation, design, and development and helps mediate between customer or consumer needs and an organizations strategic intent [5], when extended above the generalized business and technological abstraction levels, the conceptual model for IT service systems can serve the following purposes [4]: 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.

Therefore, we use the (***W4 + W4 + W14H***) specification frame of [4] for the IT-service systems Domain Requirements Modeling language. A detailed description of the (***W4 + W4 + W14H***) specification frame and its conceptual model is available in [4].

Domain requirements modeling Specification Frames

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 further 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. Also, we need a prioritization of these questions. Therefore we need an approach that allows considering the main characteristics in a systematic and surveyable way.

We discover that the specification framework may be headed by the 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). These questions generalize the Zachman framework developed far later.

Domain requirement models specify the domain's requirements in reference to a service offering

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 Modeling Language by W3C. It becomes very sophisticated with many characteristics that must be given. There is no hierarchy in the specification. An idea we might use for such characterization is *separation of*

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

concern by aspects and *layering of specifications* depending on the maturity stage. The first approach is the basis for our specification frame that allows us to describe a service. The second approach is used for the specification of an advanced cross-disciplinary IT service system.

The service is *primarily* declared by specifying

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

The ***purpose description*** governs the service. It allows 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.

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 call our framework the **W*H** (stands for ***W4 + W4 + W14H***) specification frame where H stands for How. The kernel of this framework is the (W4 + W10H) questionnaire.

5 APPLICATION OF THE SERVICE MODEL TO DOMAIN REQUIREMENT MODELING

Figure 1 (Appendix 1) is a summarized version of the application of the W*H specification frame for Domain Requirement Modeling. The fundus images that are acquired for clinical care at the institute are stored centrally in Topcon IMAGEnet i-base [16]. It provides access to those stored images from workstations around the hospital. The system contains a database of both images and patient records.

The comprehensibility of the Domain Requirement Model became the main contributor to the understanding of the domain requirements. Its compactness helps to validate domain knowledge during solution modeling discussions with the stakeholders such as Ophthalmologists with high demanding work schedules. Finally, it contribute as the primary input model for solution modeling leading to the IT-service systems projection on the evaluations criteria of systems functioning on its trustworthiness, flexibility to change, and efficient manageability and maintainability.

6 THE REFINEMENT OF THE W*H SPECIFICATION FRAMEWORK

The W*H framework provides a very convenient way of understanding a service, for learning the service, and for deployment of a service. It is based on a separation of concern into the ***end***, the ***sources***, the ***supporting means***, the ***surplus values*** of the service, the ***purpose***, the ***activities*** in the application domain, the ***parties*** involved at different phases, the ***application domain***, and the ***context***. This separation of concern provides a simple and at the same time very powerful way to understand the main ingredients of a service. We may also categorize these concerns into (a) basis and background, (b) sources and technology for development, (c) supporting means for use and the corresponding surplus values, and (d) the application domain activities. The service is additionally governed by four directives: (e) the purpose, (f) the community of practice, (g) the context of the service, and the (h) portfolio of the service. The governors are typically taken as granted. The W*H framework shows however that the governors might also be revised and evolved. The different stakeholders and parties involved form a community of practice. In Appendix II Figure 2 depicts these different concerns.

The Detailed Service Description

For service development and service use the general picture in Figure 2 (Appendix II) might be too general. We observe first that the basis and the background consist of many different aspects that have an impact on the service. The ***foundation*** is typically hidden and not visible. It consists of general paradigms that guide the corresponding discipline, of the culture as rules, standards and habits in the discipline, the business background, and finally restrictions by laws and rules that must be taken into consideration. The ***background*** of a service is given by tools that are used for the development of a service, languages which are used for description or for programming the service, and on an infrastructure that is assumed to be deployed for the service. The background depends on the community of developers and their understanding. This community has developed a number of specific pattern, styles, and routines. It has to be trained according to some given background.

Next we observe that the sources and the technology combine a large variety of different aspects for the development and construction of a service. Orchestration is typically the arrangement or composition of different features used for the service. The art of creating and arranging features into a service is often called choreography. This art rules instrumentation of features. Instrumentation is nothing else than the arrangement or composition of features. The ***development pillar*** for a service is typically based on the experiences, on the established practices and the collaboration style and pattern in the given discipline. Services use some given standards and are built on the basis of some architectures. Typically an efficient development is based on templates, configurations, and specific compositions. Development is based on systems and workbenches.

The same style of refinement may be applied to description of the supporting means and the surplus values of the service. The ***usage pillar*** of a service supports all given business cases and the envisioned usage of a service. Services are often evolving from simpler versions of the same application task and are thus consolidated.

Beside the classical supporting means we use a number of developed solutions and integrate the use of the given service with other services. The usage is harmonized with different other services by plugs. The use is guided and also supported by usage programs. Finally, the surplus value is an element of the quality characterization. We may distinguish the internal quality of a service that is a measure of the development pillar, the external quality of the service, and the quality of use which incorporates the surplus value of the service. The evaluation is bundled into a SWOT profile: Strengths, weaknesses, opportunities, and threats.

The application domain and the activities are typically based on application cases. The service satisfies a number of business requirements. The service plays a number of functions in the application.

Finally we need to revise the governors. The purpose is based on goals, on the intention of the service, and the profile of the service. The portfolio is typically based on a number of business rules.

Therefore we arrive with a more detailed model of a service depicted in Figure 3 (Appendix II).

Phases of Service Existence

Services do not occur from somewhere. We may distinguish a number of phases:

Planning: Before a system or a service is developed it must be planned. Planning takes into consideration the application cases and the envisioned functions of the service. It is governed by the business rules and the portfolio from one side and the goals, intention and profile of the service from the other side. It typically takes the foundation for granted. The background is a matter of selection and preliminary evaluation.

Building: The building phase uses the background as granted. It concentrates on the application cases. The development pillar is now the main area for concerns. These concerns are governed by the community of practice and the context of the envisioned service.

Using: After the construction and development of a service the usage pillar is the main source for concerns. The community of practice governs now the deployment, the style and pattern of use, the guideline for proper use, and the service facilities for the service. The application domain often expands the functions in which the service is going to be used. This expansion results in later requests for revision and evolution.

Evolving: Services are revised, expanded and reconsidered after some experience of use has been gained. We may distinguish between evolution, complete revision and migration. The evolution of a service results in another more powerful version of a service. Evolution does not revise the fundament of a service. It is typically also not governed by a change of the goals, intentions and the profile. All other concerns are however of interest.

These phases are based on different concerns. We concentrate during planning typically on the fundament and the options for service development. The other concerns are secondary concerns. We display the different concerns for planning, building, using and evolving in Figures 4 and 5 (Apendix II).

7 CONCLUSIONS

This research work presents two contributions: A visionary revision to medical diseases diagnosis decision support and revision to classical domain modeling in the information and service systems field by the introduction of domain requirements modeling.

Published studies of clinical decision support systems (CDSS) suggest that they enhance clinical performance for drug dosing, preventive care, and other aspects of medical care but not convincingly for disease diagnosis. The development of a DDDSN is a paradigmatic revision that enables true DDDS in the medical field. The need for the exchange of information on prevention and slowing of the disease is more dire. The autonomous nature of the medical field leads to diagnoses that are not validated by other colleagues. This practice has worked since the onset of the medical field, but a DDDSN will bring a second pair of eyes to the validation process of decisions, contributing to a more precise identification of diseases and leading to more successful treatments. The proposed development of the DDDSN is primarily for a specific disease: exudative age-related macular degeneration (ARMD). The systems evolution is estimated to accommodate 3,000 other diseases typically treated at eye hospitals. ARMD results in a deterioration of the central retinal function and is the leading cause of blindness in elderly people in Europe and the USA. The DDDSN propose to establish a network for ophthalmologists that will enable and enhance knowledge sharing for disease diagnosis decision support. The use of this network will create a shared, consistent, and scientifically accurate image repository of eye diseases for the use in diagnosis decision support.

Domain Requirement Modeling improves the understanding of the domain and its requirements acquisition for engineering cross-disciplinary IT-service systems. This novel approach fill the gap in domain specific requirements elicitation and it abstracts the service description from 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)). Domain Requirements Modeling is evaluated by application to a real-life situation – to establish a DDDSN for ARMD treatments, a true cross-disciplinary real-life application. Moreover, the models compactness helps to validate domain knowledge during solution modeling discussions with the stakeholders with high demanding work schedules. Also, Model contributes as the primary domain input model leading to the IT-service systems projection on solution modeling. In the future we will fine tune our model by introducing traceable links between the models main components and IT-service systems evaluations criteria of systems functioning on its trustworthiness, flexibility to change, and efficient manageability and maintainability

8 REFERENCES

1. Hirschheim, R., Welke, R.J. , and Schwarz, A. (2010) Service Oriented Architecture: Myths, Realities, and a Maturity Model. MIS Quarterly Executive 9(1), 204-214.
2. Spohrer, J., Maglio, P.P., Bailey, J., and Gruhl, D. (2007). Steps Towards a Science of Service Systems. IEEE Computer 40, 71-77.
3. Glushko, R.J.(2010). 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.
4. Dahanayake A., and Thalheim, B. (2012). Conceptual Model for IT-Service Systems. Journal of Universal Computer Science (Accepted for Publication)
5. Goldstein, S.M., Johnston, R., Duffy, J-A., and Rao, J. (2002). The service concept: the missing link in service design research?. Journal of Operations Management 20, 212-134, Elsevier.
6. Bjørner, D., (2006). Software Engineering 3: Domains, requirements, and software design. Springer, Berlin.
7. B. Thalheim (2009). Towards a Theory of Conceptual Modeling. In Advances in Conceptual Modeling – Challenging Perspectives. (Eds.) C. A. Heuser and G. Pernul. 1st Int. Workshop on Evolving Theories of Conceptual Modeling. pages 45-54. LNCS 5833 – Springer.
8. West, S. and Sommer, A. (2001). Prevention of blindness and priorities for the future. Bull World Health Organ, 79(3): 244-248. Retrieved from http://www.scielosp.org/scielo.php?script=sci_arttext&pid=S0042-96862001000300014&lng=en
9. Shortliffe, T.(2006). Medical Thinking Meeting, London June, Retrieved from http://www.openclinical.org/dss.html
10. Ion, A.L. and Udristoin, S. (2010). Automation of the Medical Diagnosis Process Using Semantic Image Interpretation. In Proceedings of Advances in Databases and Information Systems (ADBIS 2010), Serbia. Pp 234-246, Spinger LNCS 6295.
11. Smeulder, A.W.M., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of early years. IEEE Trans. Pattern Anal. Machine Intelligence. 22(12), pp 1349-1380.
12. Duncan, J. and Ayache, N. (2000). Medical image analysis: Progress over two decades and the challenges a head. IEEE Trans. Pattern Anal. Machine Intelligence 22(1), pp 85-106.

13. Ion,A.L(2009).Methods for Knowledge Discovery in Images. Information Technology and Control.38(1).p43-49.
14. Lui, Y., Zhang, D., Lu, G. and Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. Elsevier .Journal of the Pattern Recognition Society. 40(1). pp 262-282.
15. The Eye Disease Prevalence Research Group. (2004). Prevalence of Age-related Macular Degeneration in the United States. Arch Ophthalmol 2004; 122:564-572.
16. Topcon IMAGEnet i-base. http://www.topcon-medical.eu/eu/producten/75-imagenet-i-base.html
17. Zhou, X.S., Zillner, S., Moeller, M., Sintek, M., Zhan, Y., Krishnam, A. and Gupta, A. (2008). Semantics and CBIR: a medical imaging perspective. In Proceedings of the 2008 International Conference on Content-Based Image and Video Retrieval.
18. Deselaers, T., Keysers, D. and Ney, H. (2004). FIRE-flexible image retrieval engine: Image-CLEF 2004 evaluation. In: CLEF 2004. Springer, LNCS 3491. pp 688-698.
19. Lehnmann, T., Guld, M., Thies, C., Fischer, B., Spitzer, K., Keysers, D., Ney, H., Kohnen, M., Schubert, H. and Wein, B. (2003). The IRMA project. A state of the art report on content-based image retrieval in medical applications. In Proceedings 7th Korean- Germany Joint Workshop on Advanced Medical Image Processing. Pp 161-171.
20. Su, L., Sharp, B. and Chibelushi, C. (2002). Knowledge-based image understanding: A rule-based production system for X-ray segmentation. In Proceedings of 4th International Conference on Enterprise Information Systems. Ciudad Real, Spain. Pp 530-533.
21. Muller, H., Michoux, N., Bandon, D. and Geissbuhler, A. (2004). A review of content-based image retrieval systems in medical applications clinical benefits and future directions.International Journal of Medical Informatics.73(10) Pp 1-23.
22. Palloff, R.M. and Pratt, K. (2007). Building Online Learning Communities: Effective Strategies for the Virtual Classroom.. John Wiley & sons.
23. Thalheim, B. (2000). Entity-relationship modeling-Foundations of database technology. Springer, Berlin.
24. Dahanayake A., and Thalheim, B., (2010) Co-evolution of Information Systems Models. 10th International Conference of Exploring Modeling Methods for Systems Analysis and Design (EMMSAD), Tunisia.
25. Frakes, W. B. and Kang, K. (2007). "Software Reuse Research: Status and Future". IEEE Transactions on Software Engineering 31 (7): 529–536
26. Czarnecki, K., and Eisenecker, U.W. (2000). Generative Programming: Methods, Tools, and Applications. Boston: Addison-Wesley. ISBN 0201309777.
27. Batory, D., Johnson, C., MacDonald, B,, abd von Heeder, D. (2002). Achieving extensibility through product-lines and domain-specific languages: a case study. ACM Transactions on Software Engineering and Methodology (ACM) **11** (2): 191–214.
28. Meisinger, M.,and Rittmann, S,(2008). A Comparison of Service-oriented Development Approaches, Technical paper (TUMI-0825), Technical University of Munich, Germany.
29. Harsu, M. (December 2002). A Survey on Domain Engineering. Institute of Software Systems, Tampere University of Technology. White-paper, Report 31.
30. Kang, Kyo C.; Lee, Jaejoon; Kim, Kijoo; Kim, Gerard Jounghyun; Shin, Euiseob; Huh, Moonhang (October 2004). FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures". Annals of Software Engineering (Springer Netherlands) 5: 143–168.
31. Falbo de Almedia, R., Guizzardi, G., & Duarte, K. C. (2002). "An Ontological Approach to Domain Engineering". Proceedings of the 14th international conference on Software engineering and knowledge engineering (ACM): 351–358.
32. Dahanayake A., & Thalheim, B., (2011). “Enriching Conceptual Modeling Practices through Design Science”. 11th International Conference of Exploring Modeling Methods for Systems Analysis and Design.
33. Bjørner, D., (2009). DOMAIN ENGINEERING: Technology Management, Research and Engineering, publication of Japan Advanced Institute of Science and Technology.
34. Bergholtz, M., Andersson, and B., Johannesson, P.(2010). Abstraction, Restriction, and Co-creation: Three Perspectives on Services, ER 2010 Workshops of Conceptual Modeling of Services, LNCS 6413, 107-116.
35. Braun, P., Beeck, v-d, M., Rappl, M., and Schroder, C.(2002). Automotive Software Development: A Model-Based Approach, In-vehicle Software, SEA Technical Series.
36. Broy, M.(2005). Service-Oriented Engineering: Specification and Design of Services and Layered Architectures – The Janus Approach, In: Engineering Theories of Software Intensive Systems, 47-81, Springer.
37. Chesbrough, H., and Spohrer, J .(2006). A Research Manifesto for Service Science, CACM 49, 35-40.
38. Edvardsson, B., Gustafsson, A., and Roos, I.(2005). Service portraits in a service research: a critical review, Int. J. of Service Industry Management 16(1), 107-121.
39. Fensel, D., and Bussler, C.(2002). The Web Service Modeling Framework WSMF, Electronic Commerce: Research and Applications, 1(2), 113-137.

40. Geerts, G., and McCarthy, W.E.(1999). An Accounting Object Infrastructure for Knowledge-Based Enterprise Models, IEEE Int. C. Systems & Their Applications. 89-94.
41. Goldstein, S.M., Johnston, R., and Duffy, J-A., Rao, J.(2002) The service concept: the missing link in service design research?, Journal of Operations Management 20, 212-134, Elsevier.
42. Hurby, P.(3006). Model-Driven Design of Software Applications with Business Patterns, Springer, Heidelberg, ISBN: 3540301542.
43. IBM Research.(2004) Service Science: A New Academic Discipline?, available at http://www.almaden.ibm.com/asr/resources/facsummit.pdf.
44. Kruger, I., Mathew, R., and Meisinger, M.(2006) Efficient Exploration of Service-Oriented Architectures Using Aspects, In Proceedings of the 28th International Conference on Software Engineering (ICSE).
45. Lusch, R.F., Vargo, S.L., and Wessels, G.(2008). Toward a conceptual foundation for service science: contributions from service-dominant logic, IBM Systems Journal, 47(1).
46. McCarthy, W.E.(1982). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment, Accounting Review 57, 554-578.
47. OASIS.(2006). Reference Model for Service Oriented Architecture 1.0, http://www.oasis-open.org/committees/download.php/19679/.
48. Orfail, R., Harkey, D., and Edwards, J.(1997). Instant CORBA, Wiley.
49. Platt, D.S., and Ballinger, K.(2001). Introducing Microsoft .NET, Microsoft Press, ISBN: 0-7356-1377X.
50. Preist, C.(2004). A Conceptual Architecture for Semantic Web Services, In: Mcllraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWS 2004, LNCS-3298, 395-409.
51. Poels, G.(2010). The Resource-Service-System Model for Service Science, ER2010 Workshops, LNCS-6413, 117-126.
52. Rittmann, S.(2004). Exploring Service-Oriented Software Development for Automotive Systems, Masters' Thesis, Technical University of Munich, Germany.
53. Schatz, B., Pretschner, A. Huber, F., and Philipps, J.(2002). Model-Based Development of Embedded Systems, Technical Report (TUMI-0402), TU Munich, Germany, 2002.
54. Schewe, K-D., and 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.
55. Spohrer, J., Maglio, P., Bailey, J., and Gruhl, D.(2007). Steps Towards a Science of Service Systems, IEEE Computer, 40(1): 71-77.
56. SUN Microsystems Inc.: (2006).Java Platform, Enterprise Edition (Java EE, J2EE), Available at http://java.sun.com/javaee/, Version of Nov-15-2006
57. Vargo, S.L., Maglio, P.P., AND Akaka, M.A.(2008). On value and value co-creation: A service systems and service logic perspective, European Management Journal 26, 145-152.
58. W3C (2001): Web Description Language 1.1, Available at: http://www.w3.org/TR/wsdl. 12-Mar-2001.
59. W3C(2004). Web Service Architecture, Specifications available at: http://www.w3.org/TR/ws-arch/, 11-Feb-2004.
60. W3C Working Group.(2009). Web Service Modeling Language, Version 1.1, http://www.w3.org/TR/sml/, W3C Recommendation 12 May 2009.
61. Walsh, A.(2002). UDDI, SOAP, and WSDL: The Web Service Specification Reference book, Prentice Hall..
62. Zave, P.(2003). An Experiment in Feature Engineering, In: Annabelle McIver and Carroll Morgan, editors, Programming Methodology, pp 353-377, Springer-Verlag, New York.

Appendix 1

Service	**Disease-Diagnosis Decision Support**	**Knowledge Enhancement & Evolution**	**On-Demand Expert Learning**
(1) **End *(wherefore)***	*- Enhance Diagnosis Decision Support for ARMD*	*- Evolve with new cases and treatments*	*- Enhance specialists and residents with up-to-date diagnosis and knowledge*
(2) **Sources *(whereof)***	*- Repository of images and specific ARMD related ophthalmologic descriptions*	*- Contributions of Images from Ophthalmologists*	*- Image repository of specific ARMD related ophthalmologic descriptions*
(3) **Supporting means *(wherewith)***	*- Computerized image comparison*	*- Computerized image submission*	*- Personalized learning Wallets*
(4) **Surplus value *(worthiness)***	*- Releases ophthalmologists from tedious image by image matching*	*- Releases ophthalmologists from depending on private image collections*	*- Learning whenever necessary at own time and speed*
(5) **Purpose** • ***(Why)*** • ***(Whereto)*** • ***(when)*** • ***(for-which-reason)***	- ARMD identification *- Diagnosis* *- Early-Stage-Treatments* *- Second Opinion*	*- New knowledge* *- Diagnosis* *- Early-Treatments* *- Second Opinion*	*- Learning* *- Diagnosis* *- Early-treatments* *- Enhance knowledge*
(6) Activity • Input ***(what-in)*** • Output ***(what-out)***	*- Image of the patients eye* *-Matching image with specific diagnosis and treatment procedure descriptions*	*- Images of new cases* *- Confirmation of image submission*	*- Request for a learning Wallet* *- Personalized learning module*
(7) Party • Suppliers ***(by-whom)*** • Consumer ***(to-whom)*** • Producer ***(whichever)***	*- Ophthalmologist* *-Ophthalmologist* *-DDDS Systems*	*- Ophthalmologist* *-Image repository* *-Image repository*	*-Image Repository* *-Specialists/ internists* *-On-demand learning environment*
(8) Application domain • Application area ***(wherein)*** • Application Case ***(wherefrom)*** • Problem ***(for-what)*** • Organizational unit ***(where)*** • Triggering events ***(whence)*** • IT{data,control,comput.n} ▪ ***(what)*** ▪ ***(how)***	*-ARMD* *-During Diagnosis* *-Treatment at early stages* *-Ophthalmologic unit* *-Successful match* *-Image comparison* *-Data*	*-Maintenance* *-New Knowledge* *-Disease evolution* *-Image Repository* *-new submission* *-Knowledge Enht.* *-Data*	*-Expert Learning Evt.* *-On-demand learning* *-Personal-Wallets* *-IT Unit* *-request for learning* *-Learning Module* *-Data*
(9) Context • System context (whereat) • Story context (where-about) • Coexistence context (whither) • Time context (when)	*- Location independent Ophtalm.'s workspace* *-Impaired vision of patient* *- Integratebale Topcon IMAGEnet i-base sys.* *-On-demand*	*-Location indept. workspace* *-New knowledge submission* *-Integrated to DDDSN* *-On-demand*	*-Location indept. L-Environment* *Expert Knowledge enhancement* *-Integrated to DDDSN* *-On-demand*

Figure 1: Application of W*H Domain Requirements Modeling Framework

Appendix II

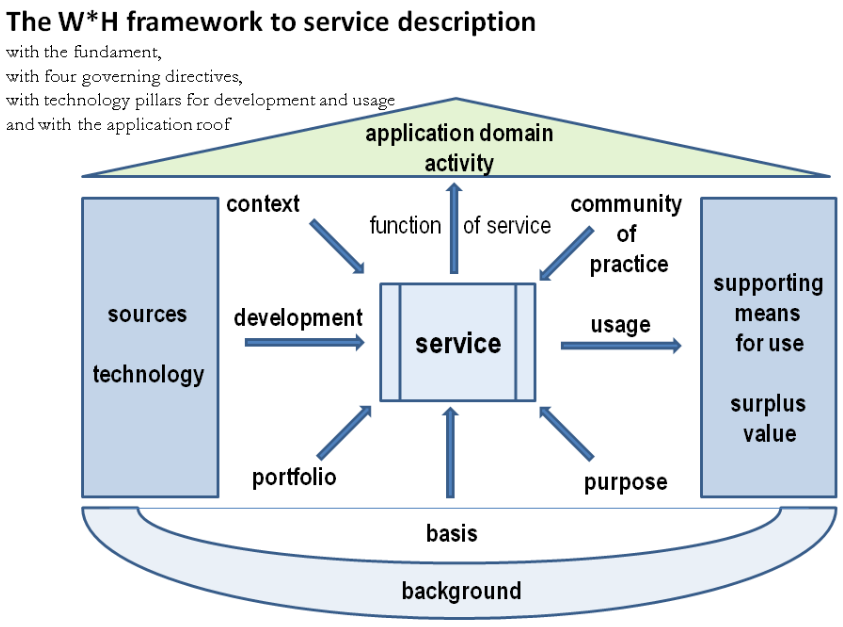

Figure 2: The Separation of Concern for the W*H Specification Framework of the Service

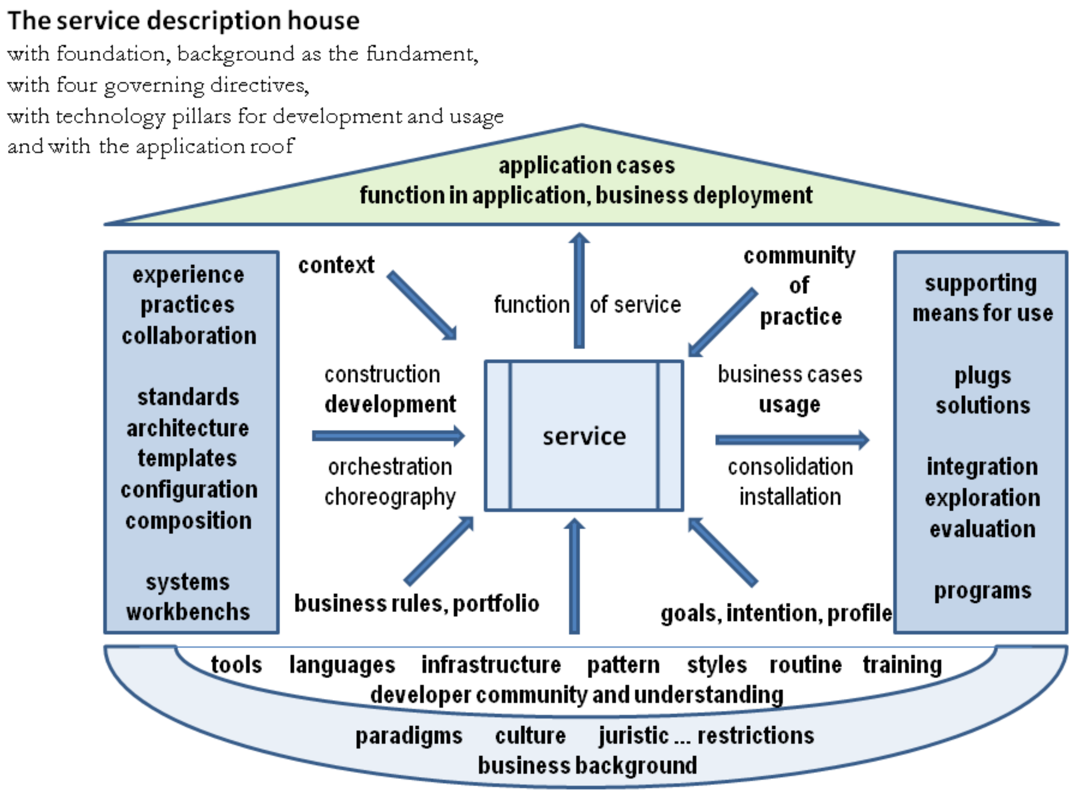

Figure 3: The Refinement of the Service Description Framework to the Service Description House

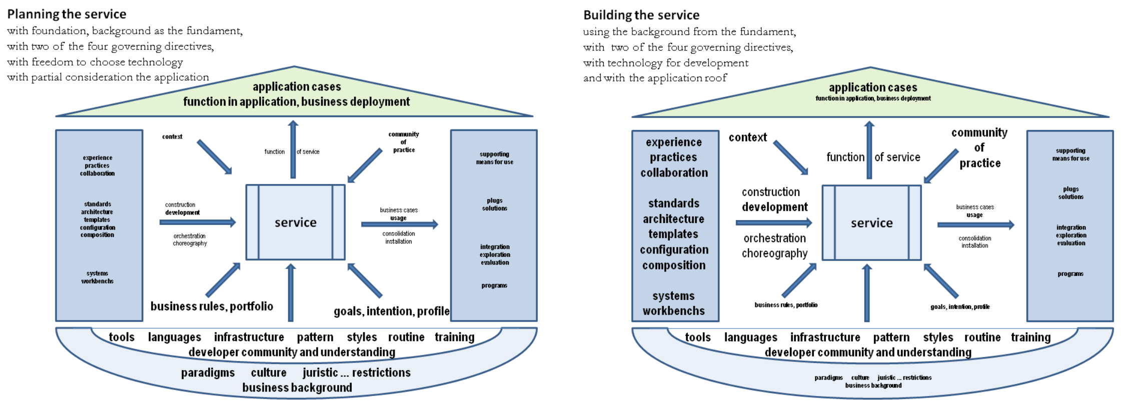
Planning the service
with foundation, background as the fundament,
with two of the four governing directives,
with freedom to choose technology
with partial consideration the application
Building the service
using the background from the fundament,
with two of the four governing directives,
with technology for development
and with the application roof
application cases
function in application, business deployment
experience practices collaboration
standards architecture templates configuration composition
systems workbenchs
context
function
of service
community of practice
construction development
orchestration choreography
service
business cases usage
consolidation installation
supporting means for use
plugs solutions
integration exploration evaluation
programs
business rules, portfolio
goals, intention, profile
tools languages infrastructure pattern styles routine training
developer community and understanding
paradigms culture juristic ... restrictions
business background

Figure 4: The Planning and the Building Phases of Service Existence

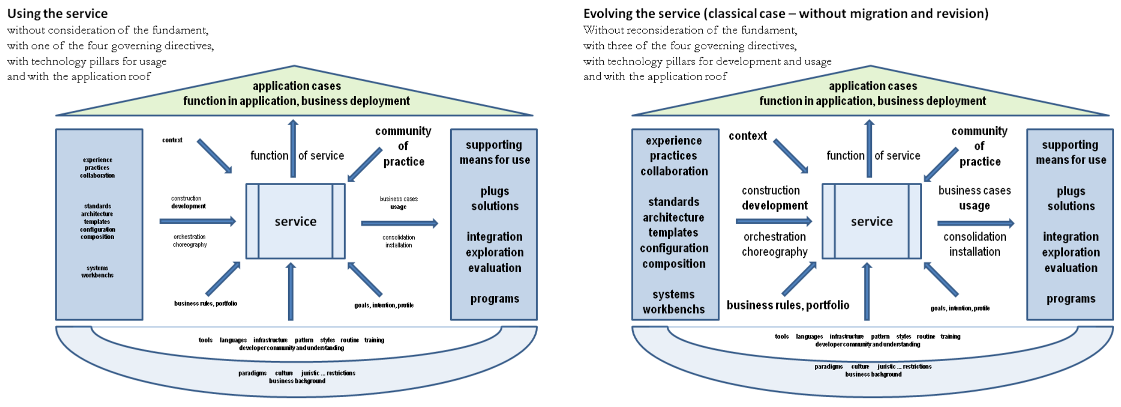
Using the service
without consideration of the fundament,
with one of the four governing directives,
with technology pillars for usage
and with the application roof
Evolving the service (classical case – without migration and revision)
Without reconsideration of the fundament,
with three of the four governing directives,
with technology pillars for development and usage
and with the application roof
application cases
function in application, business deployment
experience practices collaboration
standards architecture templates configuration composition
systems workbenchs
context
function
of service
community of practice
construction development
orchestration choreography
service
business cases usage
consolidation installation
supporting means for use
plugs solutions
integration exploration evaluation
programs
business rules, portfolio
goals, intention, profile
tools languages infrastructure pattern styles routine training
developer community and understanding
paradigms culture juristic ... restrictions
business background

Figure 5: The Using and the Evolving Phases of Service Existence