Towards an Artifact Model for Requirements to IT-enabled Product Service Systems

Marina Berkovich
Lehrstuhl für Wirtschaftsinformatik
Technische Universität München
Boltzmannstr.3
85748 Garching
berkovic@in.tum.de

Sebastian Esch
Lehrstuhl für Wirtschaftsinformatik
Technische Universität München
Boltzmannstr.3
85748 Garching
esch@in.tum.de

Christian Mauro
Lehrstuhl für Wirtschaftsinformatik
Technische Universität München
Boltzmannstr.3
85748 Garching
mauro@in.tum.de

Jan Marco Leimeister
Fachgebiet Wirtschaftsinformatik
Universität Kassel
Nora-Platiel-Str. 4
34127 Kassel
leimeister@uni-kassel.de

Helmut Krcmar
Lehrstuhl für Wirtschaftsinformatik
Technische Universität München
Boltzmannstr.3
85748 Garching
krcmar@in.tum.de

ABSTRACT

The development of IT-enabled product service systems (PSS) – a combination of physical technological elements (products) and service elements – poses various challenges because of their complexity and the involvement of multiple domains. Classical requirements engineering (RE) addresses these problems only insufficiently. This paper proposes an artifact model for the requirements to PSS, which helps in overcoming these problems. The results generated by RE or the development activities are called artifacts. The artifact model defines different types of artifacts and their interrelations. This provides a structure which facilitates the handling of a large number of requirements. The applicability of the presented artifact model is demonstrated in an example where the artifact model is applied to a real-life product. We show that the requirements can be modeled using the artifact model, and that common problems of RE can be avoided in this way.

Keywords

Product Service Systems, PSS, Hybrid Products, Requirements Engineering, Artifact Model

1. INTRODUCTION

Requirements Engineering (RE) has the task of determining correct and complete requirements [10]. RE plays an important but crucial role in the development process [10]. A poor execution of requirements engineering often results in project failures [10]. Also, defects in the product whose correction in late phases is cost-intensive are the result of poor RE [29].

Many approaches and techniques are proposed for RE in the literature. Nevertheless, RE still faces major challenges, the first of which is the communication between the participants involved in the development [11]. Especially in the development of PSS, there are different fields, ranging from marketing experts to developers, with different backgrounds and interests in the product. A common method for enhancing communication is through a medium, called artifact [12].

A second challenge in RE is the variety and complexity of the requirements resulting in difficulties structuring them [10, 11]. Stakeholders express their requirements on rather different abstraction levels. Managers, for example, think in terms of business goals and overall needs that the product has to satisfy, while operators and developers have a rather technical view, and express very concrete requirements. It is the task of RE to find the rationale for each concrete requirement by establishing a link to a higher level requirement. At the same time, the high level requirements have to be concretized to be realizable during the development [26]. Knowing the interconnections between high and low level requirements is necessary to assure the impact analyses of changes and the proper decision taking [2].

A third challenge in RE is the conceptual gap between requirements and design. RE has to support the transformation of the requirements into the design of the product [26]. This involves the so-called "translation" of the initial requirements into the "language of the developer" and the test that all requirements are correctly understood [19].

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland

These aspects of requirements engineering are especially important for complex and innovative products consisting of a high number of sub-components with a high level of technological integration. Product Service Systems (PSS) – also called hybrid product – consist of integrated bundles of physical technological components (referred to as tangible products), and intangible services [36]. By introducing PSS, companies are changing their strategy from being "product-centric" to "customer-centric" [14], i.e., they do not offer products or services, but offer solutions to customers' problems [36].

A simple example of a customer's problem is that the customer wants a constant room temperature of 21°C. He is interested in acquiring a solution for this problem as a whole, not on acquiring the single components that are necessary, such as radiator, control-software and services, e.g., maintenance [6].

The three challenges of RE mentioned above are especially important in the context of PSS [35]: (1) communication: achieving a correct and comprehensive understanding of the requirements by all domains, (2) structuring: a consistent and complete concretization and partitioning of the requirements according to the domains, and (3) an integration of the RE into the conceptual design.

This paper proposes an artifact model that addresses these issues. An artifact model provides a classification scheme for requirements and allows a problem-oriented distinction between different requirement categories [15]. It enables the stepwise concretization of the requirements in accordance with the progress of the development process and the RE. The artifact model is also a communication medium, and enhances the communication between the domains involved in the development process of PSS [12, 15]. The artifact model presented here is based on the characteristics of PSS, as well as on the insights of the role of RE in the lifecycle of PSS. The artifact model is illustrated by an example in order to demonstrate its applicability. While in this paper we focus mostly on the artifacts, the methods used to generate the artifacts and the process of applying them are mentioned only briefly.

The research presented was aligned according to design science and is explained using the guidelines of Hevner et al. [17]. The understanding of the **Problem Relevance** was done through a literature review [3, 6] and an empirical study [5]. This work resulted in a framework [4], defining that an essential part of an RE model for PSS is an artifact model, used to structure the requirements to PSS. According to the principle of **Design as a Search Process,** we regarded existing artifact models and similar concepts in our research as a background for the design of an artifact model for PSS. The principle of **Design as an Artifact** requires the result of the research to be an artifact. In our case, the developed artifact model for PSS is the artifact of our research. According to the principle of **Design Evaluation,** the artifact has to be evaluated in order to show its utility. We evaluated the artifact model by applying it to an example of real-world PSS.

2. THE ROLE OF RE IN THE DEVELOPMENT OF PSS

In the center of PSS is the idea of increasing customer satisfaction and thus generating competitive advantages [3, 7], by providing an individualized solution to the customer's problem [32]. Thus, it is important to elicit and understand the customer's requirements completely. Furthermore, the PSS is integrated both technically and organizationally into the value creation processes of the customer [14], making it necessary to understand it and derive requirements from it.

PSS integrate different components such as tangible products and services so that they are not visible to the customer particularly, but are evident as a solution [23]. A product may be hardware, software, or a combination of both hardware and software [3, 8]. The different components of the PSS are developed by product, software, and service engineering, which have different backgrounds and different understandings of the development process and requirements engineering. The domains have to be able to handle the requirements to PSS as a whole and the different components of PSS in a coordinated and complementary manner. Another aspect is modularization, meaning the fractioning of the PSS in disjunctive packages that are loosely coupled. Single modules can be standardized and reused in different PSS [8].

The lifecycle of PSS is characterized by many interdisciplinary tasks. It consists of the following phases [37]: **(1) product development**: The development phase is divided into three tasks: (a) task clarification (b) product conception and (c) development-specific component design [35]. In the first task the main parts of RE are taking place: the customer's problem is clarified and defined, and the requirements are elicited and analyzed. In this task a first decomposition of the product into tangible and intangible components is done and the requirements are partitioned accordingly. Then, in the second task, detailed function structures of the product are defined, which describe the functionality of the product. The functions are decided upon by the domain in which they are realized. Again, the requirements are partitioned according to the functions. In the third task, the single domains develop their part of the product. **(2) product marketing** and **(3) after-sales**: During these phases the requirements can change. The changes and the traceability information of changes have to be documented by the RE.

As indicated in the paragraph above, the analysis of the requirements – including their concretization and partitioning – is especially challenging for PSS. In parallel to the RE process, a conceptual and logical design of the product has to be developed [10]. This design is used to structure the requirements in a form so that they can be delivered to the development.

3. RELATED WORK

Based on an empirical study and literature reviews [3, 5, 6], we concluded that in the literature, the development of PSS (e.g. described in [23]) and also the RE are mostly elaborated upon separately. In RE no integrated handling of requirements for both products and services is present (cp. [3]).

Regarding artifact models, some related work can be found. The Requirements Abstraction Model (RAM) of Gorschek and Wohlin [16] is one of the first approaches that introduces abstraction levels for requirements. In RAM the requirements are concretized starting from high levels of abstraction to lower levels. On the higher levels the requirements are given in an abstract way, loosely defining what the product is expected to do. On the lower abstraction levels, using information of the concurrently conducted development steps, the requirements are defined in greater detail. RAM is limited to software requirements only and provides no further classification possibilities for requirements. Another artifact model is the Requirements Engineering Reference Model (REM) of Geisberger et al. [15]. The basis of their method is an artifact model that defines different classes of requirements on three abstraction levels. REM clearly focuses on software requirements for embedded systems where the hardware is already given. A third artifact model based approach is COSMOD-RE of Pohl and Sikora [30]. It is a method supporting RE in the hardware/software co-design. It distinguishes between requirement artifacts and development artifacts. The method realizes a concretization of requirements alongside the development process whereby a consolidation between the requirements and development artifacts takes place. They describe that it is important to align the requirements within the first development steps. The reviewed approaches are applied to software if hardware is given. They do not consider special topics that are important for PSS as modularization, interdisciplinarity, service requirements and hardware requirements if hardware is to be developed also.

In RE and software engineering there are many process models, e.g. [34] or V-model, but it is widely recognized that only describing the process is not sufficient. By emphasizing the results – i.e. artifacts – instead of prescribing a process, domain-specific methods for producing artifacts can be used without taking the variability of processes into account [25]. By clearly defining the artifacts to be produced, each domain involved in the PSS' development can use its special techniques or notations to develop the artifacts in a domain-specific manner. The inter-domain communication is assured by interchanging the artifact between the domains. This way, the artifacts are the basis for the inter-domain communication [21]. Because the goal of a process is always to create a result in some form, the description of the envisioned results in form of artifacts, enables the participants to focus on "what can be done", instead on "what should be done". Furthermore, precise completeness and consistency rules can be specified on artifacts easily [25].

4. REQUIREMENTS FOR AN ARTIFACT MODEL

The characteristics of PSS and the role of the RE in the lifecycle of PSS have shown that a special approach to RE for PSS is needed. In order to develop an artifact model for PSS, requirements for the model are needed. We define an **artifact** within an artifact model as a quantified information unit created or used in a development task [9]. It is a result of a development or RE activity [2]. An **artifact** bundles requirements or development information that have similar characteristics and belong to the same level of abstraction. Based on the characteristics of PSS (section 2) and the RE framework for PSS [4], the following requirements were derived.

1. **The artifact model should handle the requirements for a PSS as a whole.** PSS consist of multiple components which are not easily distinguishable. It is important to handle the requirements in an integrated manner for the whole solution [35]. The integrated handling of requirements must encompass all activities of RE, including those during the development. The artifact model must be capable of being integrated into the development.
2. **The artifact model should integrate the views of different domains**. The domains involved in the development of PSS often have different methodologies, perceptions of requirements, and understanding of the role of RE [5, 18]. It is important to handle the requirements and constraints in mutual coordination. The artifact model has to support the interdisciplinary handling of requirements and the different domain views of RE. Hence, the system behavior and the properties of the system have to be described in a form that is easily comprehensible for all involved participants.
3. **The artifact model should concretize the requirements and assign them to individual domains.** The requirements for single components of PSS have to be assigned to the responsible domains (product, software and service engineering) and to be realized using appropriate development methodologies. The development processes of the single domains take place simultaneously and in coordination [35]. The artifact model has to support this development principle by concretizing the requirements across multiple abstraction levels, as well as by assigning them to the domains and defining the interfaces necessary for the inter-domain work.
4. **The artifact model should describe relations between requirements both within one domain and between different domains.** The material and immaterial components of PSS are strongly interrelated and are hardly divisible [35]. In a holistic development approach, the interrelations between requirements must be handled independently of the domains. The artifact model must assure that the interrelations can be traced by assigning information to each artifact that describes the relationships.
5. **The artifact model should support the change management by tracing relationships.** During the development, requirements can change [34]. These changes may have effects on other requirements and on components of the system. The artifact model should realize traceability by setting the requirements in relation to each other.
6. **The artifact model should be flexible, i.e., adaptable to individual needs.** The artifact model should concretize the requirements through different levels of abstraction (proposed by [16]). Dependent on the type of PSS (whether it consists of hardware, software, services, or only two parts

of them), the needed elements have to be selected, and the necessary relations between them need to be defined.

7. **The artifact model should support module building.** PSS are structured into modules in order to enable the standardization of single parts of them [8]. The artifact model should be able to support the generation of modules.

5. AN ARTIFACT MODEL FOR REQUIREMENTS FOR A PSS

Berkovich et al. [4] are convinced that a comprehensive RE model should consist not only of a process definition and a set of techniques, but also of an artifact model. An **artifact model** provides a way of structuring and detailing the requirements step-by-step so that they can be realized by the involved domains. The development process of PSS is unique because of developing an individual solution for the customer – a solution that solves a customer's problem and integrates the elements developed by different domains. Since PSS promote an integrated and concurrent development of tangible products and services, our artifact model covers requirements to both of them.

The concepts proposed by the existing artifact models (section 3) were integrated into our artifact model for PSS. Geisberger et al. [15] first introduced the principle of structuring requirements in different artifacts. Since we propose an artifact model, this principle is the foundation of our work. However, the model of Geisberger et al. [15] has a major shortcoming: the information an artifact defines and the representation of this information are intermixed. Our model therefore explicitly discerns between representation and content of artifacts. As described by COSMOD-RE ([30]), the concretization of requirements must be integrated with the development process. It is thus necessary to establish two different viewpoints: the requirements viewpoint dealing with requirements information and the development viewpoint dealing with development information. In our artifact model, these two viewpoints are represented by two different kinds of artifacts: (a) requirements artifacts and (b) development artifacts. The concept of abstraction levels, first introduced by RAM [16], and used by Geisberger et al. [15], was incorporated in our artifact model. Our artifact model defines four abstraction levels, whereby each artifact belongs to one abstraction level.

5.1 Elements of the artifact model

In order to provide a clear structure, the meta-elements of the artifact model are described here. In Figure 1 these elements are depicted as UML class diagram. The two main types of elements in our artifact model are **abstraction levels** and **artifacts**. (The definition of an artifact was given in section 4.)

The information described by an artifact is situated at a certain level of abstraction. The **abstraction levels** divide the requirements into differently detailed layers dependent on the progress of the development process. They combine the artifacts created in the same phase of the development process, and present a layer containing requirements or development information of the same level of detail.

Apart from providing the content (of information), an artifact should also define how the information is documented. The artifact model thus explicitly separates between **model artifacts** – describing the content of information – and **representation artifacts** – describing the representation of information. This means that the model artifacts describe the content matter of an artifact, while the representation artifacts describe the type of documentation of the information.

In order to clearly distinguish between different types of artifacts, we introduce three types of model artifacts:

Requirements artifacts refer to the requirements of PSS. They are the actual work products of the RE process and support concretizing the requirements alongside the phases of RE, up to

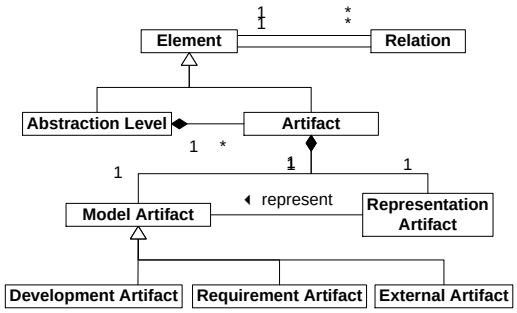

Figure 1. Elements of the artifact model for PSS

their partitioning into requirements for each domain that is involved in the development [15, 16, 29]. In order to easily address the requirements artifacts, which belong to the same level of abstraction, we bundle them into **"requirements artifacts bundles"**. **External artifacts** describe external information needed by the RE to create requirements artifacts. An example for an external artifact is a list of all stakeholders which are relevant for the development of the PSS. **Development artifacts** are work products of development tasks. Since RE and the first design steps have to be conducted concurrently, some development artifacts are needed as an input for establishing the requirements artifacts. The concurrent conduction of these two tasks enables a continuous matching of the requirements viewpoint and the design viewpoint. In this way, it is assured that the design supports the satisfaction of the requirements, and the requirements can be concretized, based on the knowledge gained by the design steps [30].

Relations between the artifacts are modeled as **Relation**. Three different types of relations are defined: (1) *concretization* meaning that a requirement is concretized by another one [15]; (2) *based-on* indicating that one artifact is created by activities that take the others as input; and (3) *impact* suggesting that one artifact is used for structuring other artifacts.

This paper presents only the model artifacts; for reasons of clarity, the representation artifacts and external artifacts are not described here. The representation of certain artifacts in the artifact model can be chosen individually, for example, depending on company-specific standards, knowledge of the participating domains, and needs of the customer.

5.2 Detailed description of the artifact model

The artifact model shown in Figure 2 consists of four abstraction levels that are based on the stages of the development process of PSS (e.g., [35, 37]) and are therefore given by the development. Each abstraction level contains artifacts, which are bundling requirements or development information having the same characteristics.

5.2.1 1st Abstraction Level – System Level

The first abstraction level is the **System Level**, consisting of requirements artifacts combined according to their similar content, and describing the generic requirements for a PSS. Based on the properties of PSS, we distinguish between four types of requirement information (cp. [3, 23]). The artifact **Customer and Stakeholder Requirements** describes the wishes of the customers [24, 29] and the requirements of other stakeholders which are relevant for the PSS to be developed. These requirements are very generic and describe the overall purpose and goals of the product. **Business Process Requirements** consist of the requirements derived from the business processes of the customer which are relevant for the PSS, since PSS are to be integrated into the value-creation process of the customer [7]. For example, if the customer wants a room temperature of 21°C, it is important to know how often the air conditioning system will actually be in operation to derive requirements for the frequency of maintenance. The artifact **Environment Requirements** describes restrictions to the realization of the PSS caused by the environment in which the PSS will be deployed. Typical environment requirements are given by laws, standards, products of competitors, technologies, development methodologies, suppliers, ecological factors, infrastructure and industry standards (e.g., [18, 24]). The artifact **Contractor's Requirements** consist of goals that the contractor wants to achieve with the PSS. They describe the resources that the contractor is able to provide for the PSS, as, for example, possible efforts to be spent. These requirements are usually the result of the abilities of the contractor [28] and the general conditions of the development process [24].

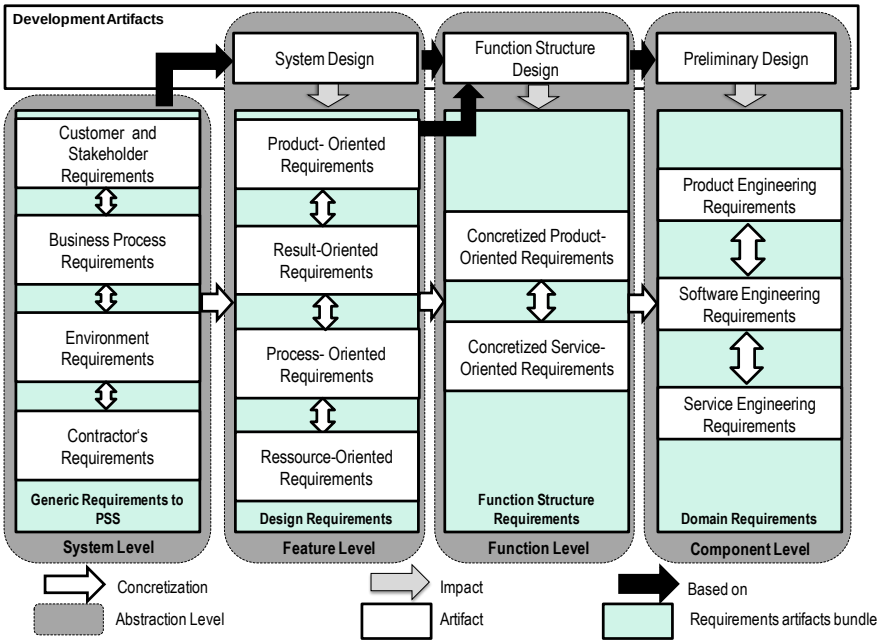

Figure 2: Artifact Model for Requirements to PSS

Summing up, the requirements of the first abstraction level describe the general requirements to the PSS on an abstract level. These requirements correspond to the definition of the initial requirements of the "task clarification" phase of the development process of PSS (see section 2).

5.2.2 2nd Abstraction Level – Feature Level

As proposed by the development process, the "task clarification" should develop a first design of the product, and decompose it into tangible and intangible parts. In our artifact model, the results of this task are stored in the 2nd abstraction level.

This abstraction level consists of a development artifact, called **System Design**, and four requirements artifacts bundled into **Design Requirements**. The system design describes the design of the product, and is generated based on the initial requirements of the 1st abstraction level. The system design defines the main functions of the PSS and decides whether they are realized by a technical product or a service.

Based on the system design, the requirements of the first abstraction level are concretized. This concretization takes the knowledge on the realization of functions, provided by the system design, into account, i.e., requirements can directly refer to the tangible product or the services which are to be developed (cp. [30]).

The system design consists of two parts: the **system boundary**, which delimits the system to be developed from other systems and defines the relation of the system to its environment [13, 27]. By defining the system boundary, the most important elements of the system and interactions with external actors are identified [30]. The second part of system design is the **function structure**, which describes the functionality of the whole PSS by means of single functions. A function is defined as the relationship of input and output parameters of a system, which serves as a purpose [31]. The communication between the different functions is described by communication paths [29]. The combination of the functions and their communication paths form the function structure and describe the entire functionality of the PSS without distinguishing the single components of it.

The functions are derived based on the requirements of the first abstraction level and the system boundary. Based on the initial requirements, the functions are concretized until it can be decided for each function whether it can be realized by a tangible product (hardware and/or software) or by a service (cp. [22, 31]). This process of concretizing the functions

is part of the development process, and is therefore not the focus of this paper, and will thus not be explained further.

The **Product-Oriented Requirements** forms an artifact and refer to the tangible components of the PSS. These requirements consider only the functionality of the tangible components, without distinguishing between hardware and software. Thus, at this stage it is still undecided of which components the products consists and how these components are realized. Only the functionality of the product is defined by these requirements. Such requirements describe, for example, the flexibility or interactions of the product with the users.

The other three requirements artifacts represent the different dimensions that are used to characterize the services [33] and to structure requirements on services [18]. The **Result-Oriented Requirements** describe the requirements to the result of a service, e.g., satisfaction of the customer with the service. The **Process-Oriented Requirements** refer to the process dimension of the services. The requirements of these artifacts describe how the process of providing the service has to be designed. The **Resource-Oriented Requirements** refer to the resources which support the provision of the services but are not the main focus of the development. An example of such a requirement is the special competence of the members of staff. Although these requirements can describe tangible products, they can be differentiated from the product-oriented requirements. They describe only resources that are needed for the provision of the service, but not the product in focus of the PSS. In other words, they describe products that are needed for the service but are not developed within the scope of the PSS.

It has to be noted that the requirements of these four types are strongly interdependent. The concretization of them takes place iteratively, whereby from each requirements artifact, requirements of all other artifacts can be derived. The result-oriented requirements are used to derive process-oriented requirements, which are used to derive resource-oriented requirements, and vice versa. These three types of requirements are the basis for the product-oriented requirements, i.e., the product-oriented requirements are derived from them [18].

5.2.3 3rd Abstraction Level – Function Level

The goal of the third abstraction level is to further concretize the functions and requirements, in order to assign later each function to a component, i.e., for each function it is decided whether it is realized by hardware (also mechatronics), software, or service. This abstraction level can be attributed to the "1a) product conception" phase of the development process of PSS (section 2). The requirements are assigned to the functions and concretized as far as necessary. We distinguish three artifacts: a development artifact **Function Structure Design** and two requirements artifacts bundled into **Function Structure Requirements**.

The process of simultaneously concretizing requirements and function structures is described by [22, 24, 31]. Here, the process is summarized, in order to explain the interrelations of the requirements and function structures. The process is conducted iteratively. As a starting point, the function structure and the requirements of the second abstraction level are taken. Then, the requirements are assigned to the functions and concretized if necessary [13, 22, 28]. In the next step, the functions are concretized and the process is repeated. This iterative concretization of requirements and functions is done until each function can be clearly assigned to one component. For each component it is decided which domain (product, software or service engineering) will realize it. The functions describing services can be concretized according to customer-involving vs. customer-neutral functions [35]. The in-depth description of this process is not the focus of this paper.

The resulting function structure consists of fine-grained functions which describe the functionalities of the technical product and services without distinguishing them in real components. For example, a function structure for the technical product washing-machine describes the complete functionality for washing like heating water, mixing the detergent, etc. A function structure for the service maintenance describes the process to provide the maintenance.

As described in the process of concretizing the function structure and the requirements iteratively, all requirements on this abstraction level are concretized and directly assigned to the functions. The **Concretized Product-Oriented Requirements** describe the technical product. The functions define the concrete functionality of the technical product and therefore the requirements of the 3rd abstraction level can be concretized in accordance with the technical characteristics like geometry, ergonomics, acoustics, user interface, etc., taking the distinction between software and hardware into consideration. The **Concretized Service-Oriented Requirements** describe the services in detail, using for example blue printing. The resource oriented requirements are described by referring to concrete resources descriptions.

The function structure describes the complete functionality of the technical product and services. At this abstraction level the realizing domain of each function is already known. Using this knowledge, the requirements are able to describe not only the functionality of the product, but also its form, e.g., its geometry.

5.2.4 4th Abstraction Level – Component Level

The fourth abstraction level concretizes and assigns the requirements to the individual domains. It therefore provides the requirements to the task "1c) development-specific component design" of the development process of PSS (see section 2).

The **Preliminary Design**, a development artifact, is a coarsely-grained description of the structure of the product under development [29]. The preliminary design is developed based on the function structure of the third abstraction level. Therefore, the product is split into hardware (also mechatronical components), software (without hardware components), and service components. The Preliminary Design concretizes the Function Structure Design and defines abstract components developed by product, software and service engineering [29]. It describes the tasks of hardware, software and services [35].

The **Domain Requirements** (requirements artifact) express requirements to the components of the preliminary design and

are a further concretization of the Function Structure Requirements. As described in the third abstraction level, all functions of the function structure are directly connected to a component of the PSS. In the fourth level of abstraction, a component for each function is defined, and the domain realizing the function is identified. The domain collects all requirements assigned to the function and concretizes them. The concretized requirements are the domain requirements of this abstraction level. The assignment of the Domain Requirements to the components of the Preliminary Design and the accompanying concretization of them is an iterative process that is described by the process model. After all requirements have been concretized, they can be divided according to functional and non-functional ones for hardware and software, and according to result-, process- and resource-oriented for services.

6. EVALUATION

The artifact model is evaluated using a criteria-based evaluation strategy according to [1]. The goal of the evaluation is to show the applicability of the artifact model on a real-life project. As a real-life example, we chose the IT-based Personal Health Manager (PHM) [20]. The PHM provides a coaching program for physical fitness to people leading an inactive lifestyle as they are either unmotivated or do not know how to do workouts. The goal of the PHM is to find the right balance between automated services that are delivered through IT, and personal services that are delivered face-to-face through coaches [20]. The idea of the evaluation is to apply the artifact model in retrospective to the requirements of the PHM. We chose an already completed project, in order to identify the occurred problems in structuring the requirements and to analyze whether these problems would be tackled by the artifact model.

6.1 Evaluation Design

The evaluation is done by assessing whether problems that occurred in the development are prevented using the artifact model.

Step 1) First, a set of criteria for the evaluation is defined. As a starting point the requirements to the artifact model (section 5) are used as criteria. Then, the developers are interviewed, to identify issues that were problematic during the development. The criteria are supplemented with these issues.
Step 2) In a joint workshop with the developers the artifact model is applied on an exemplary set of requirements.
Step 3) Then, the produced specifications are assessed by both the developers and researchers for the satisfaction of the predefined criteria. Furthermore they compared the legacy specifications and the artifact model based specification.

6.2 Evaluation Results

Step 1) In the development of the PHM a classical V-model of software engineering was applied. In a first phase, the requirements were elicited from the stakeholders and documented in a specification document. Thereby, the following list of issues occurred:

(1) Achieving consistency between requirements to services, software and hardware. This was very challenging, since, for the services, no model existed which defined how to describe the requirements and their relations to software.

(2) Achieving a consistent abstraction level of requirements and assuring the sufficient concretization of abstract requirements. The RE methods of software engineering did not provide clear criteria for the concretization of requirements, they do only state that the requirements have to be concretized till they are sufficiently detailed.

(3) Assumptions about the solution – especially which functions are realized as services and which through software – were incorporated into the specification in an unsystematic manner. Thus, rationales for the decisions were missing and it remained unclear on which information base these decisions were taken.

(4) Change-management in iterative development: Especially the requirements to services changed frequently because processes that had been performed manually, had to be automated. Thus, new requirements regarding the software came up, but the service processes changed at the same time. Both keeping an overview of the requirements and tracking changes were challenging in this setting.

(5) Incorporation of all stakeholders and sufficient requirements completeness: A large number of stakeholders with different background were involved, e.g., the users of the coaching program, the department responsible for corporate health management, or the IT service provider.

Step 2) In the workshop 20 initial requirements from the stakeholders were concretized alongside the abstraction levels of the artifact model, and resulted in 67 concrete requirements. The concretization of the requirements took place iteratively. In this paper, due to space limitations, the concretization of only one initial requirement is shown, without showing the iterations. Further, we show the concretization of just one requirement on each level. The primary stakeholders were: participants of the PHM, medical practitioners and fitness coaches, companies offering the PHM to their employees, service provider for the PHM, IT operators for the software platform, fitness studios of the companies and legislators.

***First abstraction level*:** GR1 is a *customer and stakeholder requirement* to the PSS (Table 1). The source of this requirement is the participants of the PHM. The requirement describes the high level goal that participants want to achieve.

Table 1. Requirement of the first abstraction level

Source	Requirement
Participant	**GR1:** participants should get information about physical activity and workout schedules to support them, thus becoming more active.

***Second abstraction level*:** The requirements of the first abstraction level can be concretized into Design Requirements (DR) on the second abstraction level, addressing different aspects of providing workout schedules to the participants. In Table 2, the requirements that were derived from GR1 are shown: the product-oriented requirements DR1.1 and DR1.2 and a process-oriented requirement DR1.3. First,

the system boundary was defined: all stakeholders directly communicating with the PSS are part of the system-to-be. The system was then structured into eight functions, where it was decided whether they are realized by tangible products or services. The functions are: Participant Management, Workout Supervision, Calendar Management, Content Management, Communication, Training Schedule Management, Training Course Management, and Physical Examination Management.

Table 2. Requirements of the second abstraction level

Source	Requirement
GR1	**DR1.1 (product-oriented)** A central calendar is used to manage all appointments of the participants and coaches.
GR1	**DR1.2 (product-oriented):** The workout plan must be designed so that the participant is able to increase his physical activity
GR1	**DR1.3 (process-oriented):** The workout plan is created in cooperation between the participant and the coach in order to assure that it is adequate for the participant.

***Third abstraction level*:** The eight functions of the second abstraction level describing the whole PSS were concretized iteratively, resulting in 25 functions. Here, only the functions related to the present requirements will be explained.

The requirements of the 2nd abstraction level are concretized by assigning them to the functions (Table 3). Thereby, one requirement of the 2nd abstraction level can be concretized by multiple requirements in the 3rd abstraction level. DR1.1 was concretized to FSR1.1.1 (assigned to F3: "Make appointments", and realized by the product), and to FSR1.1.2 (assigned to F2 "Appointment summary" and realized by the product).

Table 3. Requirements of the third abstraction level

Source	Function	Requirement
DR1.1	F3 "Make appointments"	**FSR1.1.1:** It must be possible to make an appointment for the creation of a workout plan, whereby coach and participant are present.
DR1.1	F2 "Appointment summary"	**FSR1.1.2:** An overview of all appointments within one month has to be provided to a participant.

***Fourth abstraction level*:** In this abstraction level, components of the PSS are defined, for which it is known whether they are realized by hardware, software, or services. Thereby, the requirements of the third abstraction level are concretized again. In Table 4 the concretization of requirements FSR1.1.1 to concrete requirements for software is shown.

Table 4. Requirements of the fourth abstraction level

Source	Component	Requirement
FSR1.1.1	*Software:* Calendar → Create Appointment	**SW1.1.1.1:** The create appointment function of the software must be able to invite both participants and coaches.
FSR1.1.1	*Hardware:* Appointment Reminder	**HW1.1.1.2:** The pulse watch must emit an acoustic signal to remind the participant of his appointment.

Figure 3 shows an excerpt of the requirements described above according to the abstraction levels, whereby the concretization relations are shown explicitly in the form of arrows.

6.3 Discussion

The artifact model provides a structure for arranging different types of requirements and for concretizing them. It defines different artifacts for services and products, and then defines the interrelations between them. Through the abstraction levels

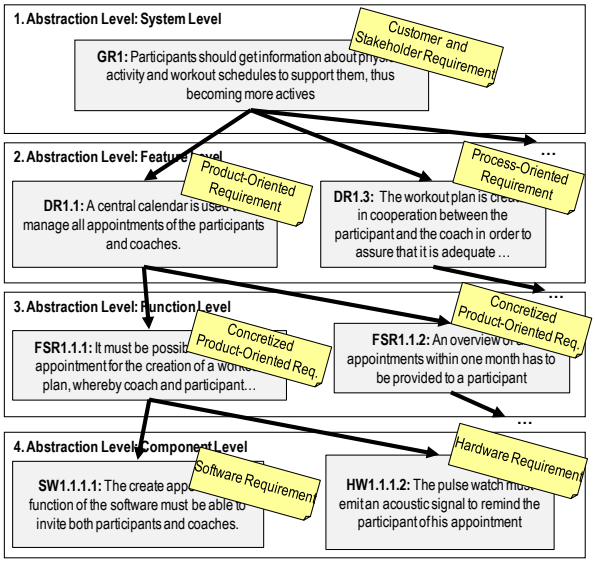

Figure 3: Application of the artifact model on the requirements to IT-based lifestyle coaching

and function structures, it defines how concretized requirements are derived from service requirements, and vice versa. Furthermore, the concretization of the requirements is aligned with the development process through the development artifacts. Thus, the co-design of requirements and development artifacts is supported. The developers noticed that the artifact model prevents an unstructured intermingling of requirements, by offering predefined categories for them. Thereby, the requirements 1 to 3 and the developer issue 1 and 2, described in section 6.2, are addressed. The incorporation of all stakeholders' requirements is facilitated by the artifact model. If a stakeholder expresses detailed requirements, they are situated on a low level of abstraction. The requirements engineer clearly sees the need to elicit high level requirements

for providing a rationale for the low level requirements. Detailed requirements are questioned and a premature focusing on realization issues is prevented. This way, in the case study the developers assigned a large number of requirements to low abstraction levels, and then recognized that high level requirements for them were missing. Thereby, they judge developer issue 2, 3 and 5 as addressed by the artifact model. At the same time, the artifact model describes general classes of artifacts and can therefore be applied to a wide range of different products, satisfying requirement 6. Through the explicit definition of artifacts for products and services on the second abstraction level, and the guidance for concretizing them in the third and fourth abstraction level, the requirements to the entire solution are specified as a whole and concretized jointly. Thereby, especially the requirements 2 and the developer issue 5 are addressed.

The artifact model defines relationships between the different artifacts. The relationships describe the interdependencies between the artifacts on the same abstraction level and the concretization dependencies between artifacts of different abstraction levels. This structuring principle supports the traceability of requirements. The requirements on higher abstraction levels serve as rationale for the requirements on lower abstraction levels. Vice versa, for each requirement on a higher abstraction level, its concretization can be found on the lower abstraction levels. The availability of this information enables efficient impact analysis when requirements change. Thereby, requirement 4 is addressed. Since traceability is a basic prerequisite for change management; requirement 5 and developer issue 4, described in section 7, are addressed. The requirements defined in the artifact model are closely aligned with the function structures. The function structures can be used to define modules. These modules can then be standardized and reused. Thereby, requirement 7 is satisfied.

6.4 Threats to Validity

The internal validity could be threatened by a bias towards the artifact model, because the developers of PHM are members of the same organization as the researchers. However, this threat is seen as minor, because the evaluation does not rely only on questioning the opinion of the developers, but their statements must be justified by the example specification. Regarding external validity, the major concern is the generalizability of the results, because we conducted only one case study. From the viewpoint of the developers of PHM and researchers, however, the selected part of the system under consideration is representative for typical projects in the field of PSS.

7. CONCLUSION

In this paper we have addressed the concept of PSS consisting of hardware, software, and service elements, offered as a bundle. Due to their special characteristics, the RE poses several challenges for them. The RE has the task of collecting and specifying all requirements on the product-to-be. Since these requirements are the base of all following development steps, they are common ground for communication and for interdisciplinary collaboration.

This paper has presented an artifact model for requirements for PSS. The artifact model defines different types of requirements – combined into artifacts – and structures them in abstraction levels. Requirements on high abstraction levels serve as rationales for requirements on lower abstraction levels. This way, it is assured that each low level requirement has a rationale, and furthermore for each high level requirements it is explicitly described which low level requirements realize them. Thus, the completeness of low level requirements is increased and traceability between these requirements is realized.

Another distinguishing mark of the artifact model is the integration of the development artifacts into the RE. The importance of relying on initial design decision for concretizing requirements has been acknowledged in the RE in software engineering. Through the explicit modeling of the dependencies of development artifacts and requirements artifacts, a concerted concretization and structuring of requirements is enabled. This way, it is avoided that preliminary design decisions are incorporated into the specification unknowingly and in an unstructured manner. By focusing on the artifacts instead of processes, the inter-domain cooperation is enhanced. By clearly defining the artifacts to be produced, each domain can use its special tools, notations and techniques to develop the artifacts in a domain-specific manner. Additionally, the artifacts are the basis for the inter-domain communication.

Entirely new in the proposed artifact model is the combination of requirements for all components of PSS: Software-, hardware-, and service requirements are handled using one comprehensive artifact model. It therefore serves as a common basis for the understanding of all participating domains and for communication during development activities.

The applicability of the artifact model has been illustrated by a real-life example. In cooperation with the initial developers of the example system, the satisfaction of the requirements has been discussed. Further, five major problems experienced during the development have been tackled. Thus, we conclude that the artifact model is applicable in practice and helps addressing common problems.

7.1 Limitations and Future Work

A limitation of this work is that the evaluation was only conducted in retrospective. However, this way it was possible to compare the problems experienced during the development, with the benefits the artifact model could provide. Another limitation is that due to space restrictions the representation of the artifacts' content, the process model, and the techniques for creating the artifacts could not be described. Further research will focus on more comprehensive case studies to show the usefulness of the artifact model. In order to conduct such case studies, a process model and a set of methods have to be elaborated upon. A tool support for the artifact model would be beneficial as well, since in real-life projects a large number of requirements have to be managed.

8. ACKNOWLEDGEMENT

We thank the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) for funding this project as part of the collaborative research center „Sonderforschungsbereich

768 – Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products".

9. REFERENCES

[1] Ahlemann, F. and Riempp, G. 2008. RefModPM: A Conceptual Reference Model for Project Management Information Systems. In *Wirtschaftsinformatik 50* (2).

[2] Berenbach, B., Paulish, D.J., Kazmeier, J. and Rudorfer, A. 2009. *Software & Systems Requirements Engineering: In Practice*. Mcgraw-Hill Professional.

[3] Berkovich, M., Esch, S., Leimeister, J.M. and Krcmar, H. 2009. Requirements engineering for hybrid products as bundles of hardware, software and service elements – a literature review. In *Proceedings of the 9. Internationale Tagung Wirtschaftsinformatik* (Wien, Österreich, 2009).

[4] Berkovich, M., Leimeister, J.M. and Krcmar, H. 2010. Ein Bezugsrahmen für Requirements Engineering hybrider Produkte. In *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2010)* (Göttingen, 2010).

[5] Berkovich, M., Leimeister, J.M. and Krcmar, H. 2009. An empirical exploration of requirements engineering for hybrid products. In *Proceedings of the XVIIth European Conference on Information Systems* (Verona, 2009).

[6] Berkovich, M., Leimeister, J.M. and Krcmar, H. 2009. Suitability of Product Development Methods for Hybrid Products as Bundles of Classic Products, Software and Serivce Elements. In *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE* (San Diego, 2009).

[7] Böhmann, T. and Krcmar, H. 2007. Hybride Produkte: Merkmale und Herausforderungen. In *Wertschöpfungsprozesse bei Dienstleistungen: Forum Dienstleistungsmanagement*, Bruhn, M. and Stauss, B. Ed. Gabler, 240-255.

[8] Böhmann, T., Langer, P. and Schermann, M. 2008. Systematische Überführung von kundenspezifischen IT-Lösungen in integrierte Produkt-Dienstleistungsbausteine mit der SCORE-Methode. In *Wirtschaftsinformatik, 50* (3).

[9] Booch, G., Rumbaugh, J. and Jacobson, I. 2007. *Das UML Benutzerhandbuch*. Addison-Wesley, München.

[10] Byrd, T.A., Cossick, K.L. and Zmud, R.W. 1992. A synthesis of research on requirements analysis and knowledge acquisition techniques. In *MIS Quarterly*, *16* (1), 117-138.

[11] Davis, G.B. 1982. Strategies for information requirements determination. In *IBM Systems Journal*, *21* (1), 4-30.

[12] Dix, A.J. 1994. Computer-supported cooperative work - a framework. In *Design Issues in CSCW*, Rosenburg, D. and Hutchison, C. Ed. Springer Verlag, 23-37.

[13] Ehrlenspiel, K. 2002. *Integrierte Produktentwicklung*. Hanser Fachbuchverlag.

[14] Galbraith, J.R. 2002. Organizing to Deliver Solutions. In *Organizational Dynamics*, *31* (2), 194-207.

[15] Geisberger, E., Broy, M., Berenbach, B., Kazmeier, J., Paulish, D. and Rudorfer, A. 2006. *Requirements Engineering Reference Model (REM)*. Technical Report. Technische Universität München, München.

[16] Gorschek, T. and Wohlin, C. 2006. Requirements Abstraction Model. In *Requirements Engineering*, *11* (1).

[17] Hevner, A.R., March, S.T., Park, J. and Ram, S. 2004. Design Science in Information Systems Research. In *MIS Quarterly*, *28* (1), 75-105.

[18] Husen, C.v. 2007. *Anforderungsanalyse für produktbegleitende Dienstleistungen*. Doctoral Thesis. Fakultät Maschinenbau, Universität Stuttgart.

[19] Jacobson, I., Booch, G. and Rumbaugh, J. 1999. *Unified Software Development Process: The complete guide to the Unified Process from the original designers*. Addison-Wesley Longman, Amsterdam.

[20] Knebel, U., Esch, S., Leimeister, J.M., Pressler, A. and Krcmar, H. 2009. Online, Set, Go - Design and empirical Test of an Itbased physical Activity Intervention. In *Proceedings of the 17th European Conference on Information Systems*, Verona.

[21] Kofler, T. and Ratiu, D., Towards a Reusable Unified Basis for Representing Business Domain Knowledge and Development Artifacts in Systems Engineering. In *Proceedings of the Workshop on Advances in Conceptual Modeling*, Vancouver.

[22] Kortler, S., Helms, B., Berkovich, M., Lindemann, U., Shea, K., Leimeister, J.M. and Krcmar, H. 2010. Using mdm-methods in order to improve managing of iterations in design processes. In *Proceedings of the 12th International dependency and structure modelling conference, DSM*, Cambridge.

[23] Leimeister, J.M. and Glauner, C. 2008. Hybride Produkte – Einordnung und Herausforderungen für die Wirtschaftsinformatik. In *Wirtschaftsinformatik, 50* (3).

[24] Lindemann, U. 2006. *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden* Springer, Berlin.

[25] Méndez Fernández, D., Penzenstadler, B., Kuhrmann, M. and Broy, M., A Meta Model for Artefact-Orientation: Fundamentals and Lessons Learned in Requirements Engineering. In *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems*, Oslo.

[26] Nikora, A.P., Classifying requirements: towards a more rigorous analysis of natural-language specifications. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*.

[27] Nuseibeh, B.A. and Easterbrook, S.M., Requirements Engineering: A Roadmap. In *Proceedings of the 22nd International Conference on Software Engineering*.

[28] Pahl, G., Beitz, W., Feldhusen, J. and Grote, K.-H. 2006. *Engineering Design: A Systematic Approach*. Springer, Berlin.

[29] Pohl, K. 2007. *Requirements Engineering. Grundlagen, Prinzipien, Techniken.* Dpunkt Verlag.

[30] Pohl, K. and Sikora, E., COSMOD-RE: Supporting the Co-Design of Requirements and Architectural Artifacts. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*.

[31] Ponn, J. and Lindemann, U. 2008. *Konzeptentwicklung und Gestaltung technischer Produkte: Optimierte Produkte - systematisch von Anforderungen zu Konzepten*. Springer, Berlin.

[32] Sawhney, M. 2006. Going beyond the Product: Defining, Designing and Devlivering Customer Solutions. In *The Service-dominant Logic of Marketing,* Lusch, R.F. and Vargo, S.L. Ed., M. E. Sharpe, New York, 365-380.

[33] Scheer, A.-W., Grieble, O. and Klein, R. 2003. Modellbasiertes Dienstleistungsmanagement. In *Service Engineering - Entwicklung und Gestaltung innovativer Dienstleistungen,* Bullinger, H.-J. and Scheer, A.-W. Ed., Springer, Berlin.

[34] Sommerville, I. and Kotonya, G. 1998. *Requirements Engineering: Processes and Techniques* Wiley & Sons.

[35] Spath, D. and Demuß, L. 2003. Entwicklung hybrider Produkte – Gestaltung materieller und immaterieller Leistungsbündel. In *Service Engineering - Entwicklung und Gestaltung innovativer Dienstleistungen,* Bullinger, H.-J. and Scheer, A.-W. Ed., Springer, Berlin.

[36] Tan, A.R., McAloone, T.C. and Gall, C., Product/Service-System Development - An explorative Case Study in a manufacturing Company. In *Proceedings of the international conference on engineering design,* Paris.

[37] Tuli, R., Kohli, A. and Bharadwaj, S. 2007. Rethinking Customer Solutions: From product Bundles to Relational Processes. In *Journal of Marketing, 71 (3).*