Specifying a modelling language for PSS Engineering – A development method and an operational tool

Khaled Medini, Xavier Boucher

Version of Record: https://www.sciencedirect.com/science/article/pii/S0166361518306547
Manuscript_579d815f92d238b7372f5437e1cdfb02

Specifying a modelling language for PSS engineering – a development method and an operational tool

Khaled Medini[a*], Xavier Boucher[a]

[a]*Mines Saint-Etienne, Univ Clermont Auvergne, CNRS, UMR 6158 LIMOS, Henri Fayol Institute, F - 42023 Saint-Etienne France*

*Corresponding author
Address: Mines Saint-Etienne, 158 Cours Fauriel, 42023 Saint-Etienne, France
Email: khaled.medini@emse.fr
Phone: +33 4 77 42 93 17

Abstract. Although the literature is full of research on the transition of industry towards Product-Service Systems (PSS), the question of how to effectively support PSS engineering is poorly addressed. The compelling need for decision support throughout the various stages of the engineering process is particularly challenging due to the inherent complexity of PSS. In this sense, visualisation and modelling at large have been put forth as promising means for supporting PSS engineering. This paper proposes a method for specifying a modelling language for PSS engineering, putting together PSS domain-specific knowledge and modelling concepts inherited from conceptual modelling and model-based engineering. It relies on a recursive transformation process of the underlying PSS meta-model using knowledge from case studies and the literature. The method has proven to be a practical means for gradual enrichment of the modelling language leading to successful experimentations in the industrial context.

Keywords: PSS engineering, conceptual modelling, modelling language, domain-specific modelling, model-based system engineering.

Specifying a modelling language for PSS Engineering – a development method and an operational tool

Abstract. Although the literature is full of research on the transition of industry towards Product-Service Systems (PSS), the question of how to effectively support PSS engineering is poorly addressed. The compelling need for decision support throughout the various stages of the engineering process is particularly challenging due to the inherent complexity of PSS. In this sense, visualisation and modelling at large have been put forth as promising means for supporting PSS engineering. This paper proposes a method for specifying a modelling language for PSS engineering, putting together PSS domain-specific knowledge and modelling concepts inherited from conceptual modelling **and model-based engineering**. It relies on a recursive transformation process of the underlying PSS meta-model using knowledge from case studies and the literature. The method has proven to be a practical means for gradual enrichment of the modelling language leading to successful experimentations in the industrial context.

Keywords: PSS engineering, conceptual modelling, modelling language, domain-specific modelling, model-based system engineering.

1. Introduction

The industrial transition towards more customer-centric operations, further automation and increasing digitalization has fostered the development of new business models such as Product-Service **Systems** (PSS). PSS can be seen as an integrated offering consisting of products and services to meet individual customer expectations. While PSS have been widely adopted in many sectors, PSS engineering is still hindered by several challenges **owing to its inherent complexity**. **PSS complexity stems from combining (tangible) products and (intangible) services within a unique offering. This adds to the complexity of the cross-domain PSS engineering process, calling for various skills. Furthermore, changing customer requirements suggest a close collaboration among PSS stakeholders (including customers) in order to continuously adapt the PSS (Berkovich et al., 2011; Cavalieri and Pezzotta, 2012; Vasantha et al., 2012; Boucher et al., 2016; Wolfenstetter et al., 2018). Subsequently, proper methods and tools are needed to mitigate the engineering process complexity. Hence recent research has started to apply the System Engineering (SE) principles to PSS engineering, for instance (Trevisan and Brissaud, 2016; Wiesner et al., 2017; Maleki et al. 2018). System engineering is defined by the International Council on Systems Engineering (INCOSE) as *an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer and stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire life cycle*. However, viewing PSS as a system and using SE principles adds other challenges to the PSS engineering process, due to the significant amount of knowledge to be managed (PSS requirements, PSS structure, PSS behaviour, life cycle perspective, etc.).**

PSS visualisation has been recognised as an important means for elucidating knowledge throughout PSS engineering (**Lee and Kim, 2010; Geng and Chu, 2011;** Cavalieri and Pezzotta, 2012; Lim et al., 2012; Bertoni et al., 2013; **Wolfenstetter et al., 2018**). **The elucidation process allows for sharing a**

common understanding among the PSS stakeholders of the whole PSS including the functional domain (value expectation) and solution domain (value proposition) (Meier et al., 2010; Wolfenstetter et al., 2018). The prerequisite of the elucidation process is to enable smooth integration of different domain-specific models into a holistic and comprehensive PSS representation.

To cope with such integration, conceptual modelling as a (formal) representation of systems has been identified as one major method which can be used for developing models, languages and methods easing knowledge sharing across multi-disciplinary designers or researchers (Mylopoulos, 1992; Karagiannis and Heinrich, 2016; Bock et al., 2017).

This paper proposes a method for specifying a modelling language for PSS engineering, putting together PSS domain-specific knowledge and modelling concepts inherited from conceptual modelling and model-based engineering. The method has proven to be a practical approach for gradual enrichment of the modelling language. It relies on a recursive transformation process of the underlying meta-model, integrating knowledge extracted from several case studies and the literature.

The remainder of the paper is organised as follows: section 2 provides a state of the art focusing on PSS engineering and providing basic modelling concepts for the method proposed. Section 3 elaborates a method for building a PSS modelling language. Section 4 applies the method within an industrial context, based on knowledge extraction from several case studies. The resulting modelling language and its implementation are presented in Section 5. Section 6 discusses the proposed method. Concluding remarks are presented in Section 7.

2. State of the art and research method

2.1. PSS engineering complexity and the need for visualisation methods

PSS engineering has received a great deal of attention from the scientific community due to its inherent complexity. The l**ast decade witnessed an increasing number of publications dealing with PSS conceptualisation and PSS engineering (Meier et al., 2010; Berkovich et al., 2011; Cavalieri and Pezzotta, 2012; Beuren et al., 2013; Wiesner et al., 2015; Qu et al., 2016; Marques et al., 2017). However, while much of this research work argues about the multidimensional intrinsic nature of PSS, operational guidance for PSS engineering is still scarce, owing to the lack of tools supporting PSS understanding and evaluation (Meier et al., 2010; Vasantha et al., 2012; Beuren et al., 2013; Wolfenstetter et al., 2018).**

Unlike traditional products, PSS design calls for a cross-domain engineering process relying on various backgrounds related to product and/or service design (Vasantha et al., 2012; Wolfenstetter et al., 2018). This comes with a major challenge: how to integrate the contributions of engineers/stakeholders from different backgrounds into the design of a successful PSS. Most of the models used by the engineers are domain-specific (CAD models, service blueprints, simulation models, etc.) each representing a single perspective of the PSS. This is likely to impede the communication and the collaborative development of the PSS (Wiesner et al., 2017; Wolfenstetter et al., 2018). Furthermore, changing customer requirements calling for PSS customisation require good traceability allowing for easy mapping of the PSS components in the requirements (Meier et al., 2010; Marques et al., 2017; Hribernik et al., 2017; Khan and Wuest, 2018; Wolfenstetter et al.,

2018). Subsequently recent works have started to promote system engineering as a means to foster PSS engineering and thus development in the manufacturing sector. This is consistent with the nature of PSS which is intrinsically a system and needs to be addressed throughout its life cycle (Cavalieri and Pezzotta, 2012; Trevisan and Brissaud, 2016; **Wiesner et al., 2017;** Maleki et al. 2018). While SE was originally intended for designing complex technological systems, its potential in the PSS domain has been made apparent in several research works. Trevisan and Brissaud (2016) developed an integrated modelling framework aimed at putting together existing models from product design and service design domains. Maleki et al. (2018) proposed a PSS meta-model to support PSS design following the System Engineering approach. In the proposed meta-model, both the PSS and its supporting system are modelled considering a life cycle perspective.

As a corollary to this, **visualisation as a means to graphically represent PSS, is put forth** as a promising way to mitigate design complexity and support communication among PSS stakeholders (Sakao and Shimomura, 2007; Geum and Park, 2011; Vasantha et al., 2012; Bertoni et al., 2013; Qu et al., 2016). This is evidenced by the significant piece of literature dealing with PSS visualisation (Lim et al., 2012; Bertoni et al., 2013; Durugbo et al., 2011).

Although several works have contributed to advancing the research in the PSS visualisation domain, **a comprehensive approach covering PSS dimensions is still missing as the focus has often been on a particular PSS aspect, e.g. service activities (Sakao and Shimomura, 2007; Geng and Chu, 2011; Geum and Park al., 2011), requirements and PSS functions (Lee and Kim, 2010), PSS life cycle perspective (Wiesner et al., 2015; Wellsandt et al. (2017**). **Therefore, there is a compelling need for methods and tools to support PSS visualisation along the engineering process. The ultimate goal is thus to enable more informed decisions and to build a common understanding of the PSS envisioned.** Furthermore, most of the contributions **available** provide evidence of the applicability of the visualisation **techniques**; however the elaboration methods of the visualisation techniques and modelling language are usually implicit. The objective of the current paper is to frame a comprehensive PSS modelling language and to specify the elaboration method thereof.

Building on the inherent complexity of PSS and the underpinning system view, the next section gives an overview of the modelling concepts and principles which provides the foundations for the specification and implementation of a PSS modelling language supporting its engineering process.

2.2. Conceptual modelling and modelling concepts

The aim of this section is not to perform an exhaustive literature review on modelling, but merely to introduce relevant modelling concepts supporting **the formalisation of a new PSS** modelling**.** In this sense, conceptual modelling was identified as an appropriate context for discussing modelling concepts. The appropriateness of conceptual modelling to large-scale and complex modelling and simulation problems has fostered its use among various communities (Karagiannis et al., 2016; Bock et al., 2017). Conceptual modelling can be defined as *the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication* (Mylopoulos, 1992). Conceptual modelling is closely related to formal representations of systems which supports integration across engineering domains (Klinger and Becker, 2012; Bock et al., 2017). **According to Becker et al. (2010), conceptual modelling eases the communication among interdisciplinary project teams due to its formality in representing a system.** This characteristic is

consistent with PSS engineering which involves cross-disciplinary teams with backgrounds in product design, service engineering, management, marketing, etc.

It can be inferred unambiguously from the Mylopoulos definition that conceptual modelling relies on representing systems using models. Jouault et al. (2008) see a model as a representation of a system which captures some of its characteristics and provides knowledge about it. Complementarily, Silva (2015) defines a model as an abstraction of a system under study which may already exist or is intended to exist in the future. Selic's (2003) definition introduces the viewpoint of models for understanding a system; he sees a model *as a reduced representation of some system that highlights the properties of interest from a given viewpoint*. Models support the definition of a common vision and sharing knowledge among different stakeholders. Model definitions and characteristics including abstraction, knowledge sharing and viewpoints, are consistent with PSS peculiarities, in particular the need to visualise such a complex system in a way to support its engineering process.

Usually, models are formally developed using modelling languages. The latter can be defined as *a set of possible models that are conformant with the modelling language abstract syntax, represented by one or more concrete syntaxes and that satisfy a given semantics* (Silva, 2015). The abstract syntax refers to a meta-model which is also referred to as a model of a model. Silva (2015) defines a meta-model as a model that defines the structure of a modelling language. Concrete syntax refers to the graphical notations used by the modelling language. Depending on the modelling language scopes, these can be General Purpose Modelling Languages (GPML) or Domain-Specific Modelling Languages (DSML). A GPML, as a set of generic modelling concepts, is assumed to be independent from any particular domain. A commonly used example of GPML is the Unified Modelling Language (UML). Adversely, a DSML enriches modelling concepts with domain-specific knowledge and is relevant to such specific domains (Frank, 2010).

Klinger and Becker (2012) introduced a formal modelling approach to represent product and service portfolios. They highlight the need for formally representing the interdependencies between product and service portfolios (inter-tree dependencies) and within product and service portfolios (intra-tree dependencies). The resulting notations are expected to support PSS configuration. Becker et al. (2010) investigated the potential of existing conceptual (reference) models to PSS representation. They came to the conclusion that several constructs embedded in the models available fit into some areas in the PSS domain. However their analysis uncovered several shortages of existing models, namely, integration of products and services, and inappropriateness of service-oriented models to product and vice versa. Becker et al. (2010) argue that one way of addressing these issues is to integrate modelling languages. This could be enabled through merging the meta-models of different modelling languages into a new meta-model and then instantiating conceptual models that are compliant with the new meta-model (Becker et al., 2010).

In light of the gaps identified in the literature, the aforementioned concepts will be useful for both the process for building the PSS modelling language (section 3) and specifying the modelling language (sections 4 and 5).

2.3.Research method

The objective of the research method is to specify a PSS modelling language while building on case studies and literature In case studies *researchers develop an in-depth analysis of a case, often a program, event, activity, process, or one or more individuals* (Creswell, 1994). This choice is due to the lack of integrated models covering the various PSS dimensions, also illustrated in the literature review findings. The aim is thus to explore use cases in order to derive concepts relevant to the current study, following an iterative process defined in section 3. A multiple case study was chosen for the sake of robustness of the results. Data from the case studies are collected through interviews, work meetings and reports. These data support a progressive specification of the modelling language. In order to ensure more consistency, the output from the case studies is confronted to reference models from the literature so as to check, improve and/or validate it. Complementarily, a final validation of the industrial applicability consists in experimenting the modelling language in an industrial context, as reported in section 5.2. This subsequently allows for extension of the body of knowledge in the literature on PSS engineering with empirical findings.

3. Iterative process for building a modelling language for PSS engineering

3.1.Overview of the method

This section elaborates on a method for developing a DSML supporting PSS engineering. The proposed modelling method is intended to support the PSS representation in a comprehensive way by covering different perspectives of the PSS engineering scope. The ultimate objective is to enable PSS representation so as to support the communication and sharing of a common understanding of the PSS solutions by the various stakeholders, throughout the engineering process. The method relies on an iterative process driven by industrial PSS case studies and reference models from the literature. The process basically consists in gradual enrichment of meta-models with knowledge from use cases and existing literature. Figure 1 shows the inputs, outputs, controls and mechanism of the elementary activity of the iterative process, following the IDEF0 standards (National Institute of Standards and Technology, 1993). For a set *n* of use cases, the iterative process consists of a series of *n* elementary processes namely, Process use case *i*, with *i* ranging from *1* to *n*. Process use case *i* consists in applying an initial meta-model *i-1* (the input) to a given use case *i*, in order to extend the scope of the meta-model *i-1*. The output is an extended meta-model *i*.

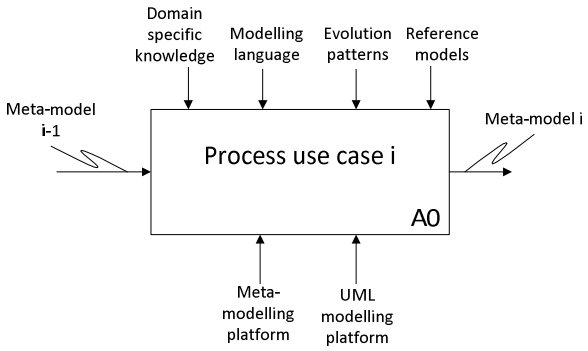


Figure 1: Elementary activity of the iterative process

The transformation process impacting factors are represented by the controls of the elementary process (Figure 1) namely, domain-specific knowledge, modelling language, reference models, and evolution patterns. The iterative process uses UML – Unified Modelling Language in order to formally represent knowledge from use cases. UML is a general purpose and object-oriented modelling language, whose specifications are detailed by the Open Management Group (OMG) (OMG, 2017; Silva, 2015). The domain-specific knowledge is basically built upon data collected from use case reports and information system software tools and through interviews with personnel from the use cases. The reference models refer to PSS models existing in the literature which may be used to extend the meta-model **(e.g. Lim et al., 2012). In the event that no explicit models are discussed, relevant PSS-related concepts may be derived from conceptual or review papers (Cavalieri and Pezzotta, 2012)**. The extension is enabled by an iterative transformation of the meta-models supported by the evolution patterns. These patterns can be defined as a set of rules for adapting a given (meta-) model towards more generic ones with extended scopes. Three main patterns are used namely generalisation, specialisation and grouping. Generalisation consists in identifying a new concept embedding common inherent properties of different concepts. Specialisation refers to breaking down a general concept into several concepts, each reflecting a specific piece of domain-specific knowledge. Grouping consists in creating a pool of concepts sharing common properties in a given context. **The basic activity of the proposed method namely '*Process use case i*' (see Figure 1) is detailed in Section 3.3, titled model transformation process. For the sake of readability, the supporting modelling platforms are presented beforehand, in Section 3.2.**

3.2.Supporting modelling platforms

The supporting platforms are represented by the mechanisms of the elementary process (bottom arrows). The process is supported by a UML modelling platform and a meta-modelling platform. The former allows for building and updating meta-models while the latter supports the implementation of the meta-models and the application to use cases.

To illustrate the subsequent steps (i.e. details of the iterative process and model transformation), it is worthwhile providing an overview of the software platforms supporting meta-model implementation and application to use cases. To build and update meta-models, the Eclipse Luna platform was selected and a plug-in to build UML class diagrams consistently with OMG standards was installed. For meta-model implementation and application, an open meta-modelling platform was selected, namely ADOxx. This platform is consistent with a meta-modelling approach that was developed within the framework of the Open Models Laboratory (OMiLAB) (Karagiannis et al., 2016). The abstraction layers supporting the meta-modelling approach are shown in Figure 2.

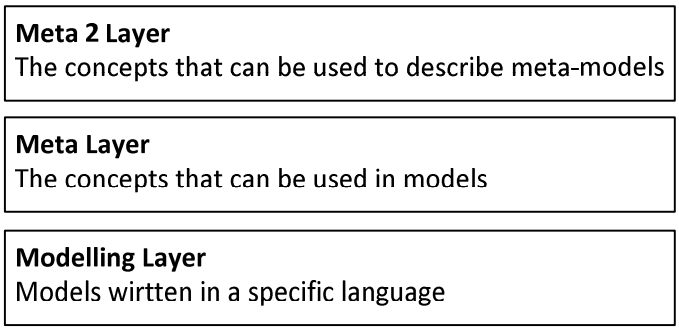


Figure 2: Abstraction layers in meta-modelling, adapted from (Karagiannis et al., 2016)

The ADOxx platform provides a powerful development and modelling toolkit enabling, respectively, the implementation of a given meta-model in a library and the use of this library as a modelling tool to instantiate the meta-model using use cases. Unlike most of the meta-modelling platforms, ADOxx provides very useful graphical modelling facilities. Additionally it allows a smooth back and forth process for updating the meta-model layer based on requirements from the modelling layer and vice versa.

3.3.Model transformation process

Figure 3 shows the details of the elementary process described above. For a given use case *i*, the transformation process starts with modelling the PSS scenario using an UML modelling platform (step A0-1), resulting in a draft meta-model *i*. Reference models from the literature can be used as a starting point or to enrich the meta-model if additional concepts (or associations) are needed. The basic tool for evolving a meta-model *i-1* is evolution patterns, which are applied according to the need of use case *i* beyond meta-model *i-1*.

The resulting draft meta-model *i* is then implemented in the ADOxx platform (step A0-2). The implementation consists in transforming the meta-model described using UML class diagrams into an ADOxx library. Model transformation is quite straightforward as the library relies on object-oriented modelling. UML classes and associations are respectively represented by class objects and association class objects within the meta-modelling library. Additional attributes are available in the ADOxx library for the class object for graphical and calculation purposes. The different association types in UML can be modelled using constraints in the ADOxx library.

The output of step A0-2 is an ADOxx library of the draft meta-model specifying a modelling language. The subsequent step, A0-3, consists in building an instance of use case *i* using the ADOxx platform. In other words, use case *i* is modelled consistently with the draft meta-model *i* using the ADOxx library. The instance resulting from step A0-3 undergoes a consistency check in step A0-4, in order to see whether the concepts used in modelling the instance are still consistent with meta-model *i-1*. This ensures a gradual extension of the scope of the method without losing consistency. If the consistency check suggests that refinements are required, then an update of use case *i* meta-model is needed (step A0-5). Similarly to step A0-1, the refinement uses evolution patterns. The result of step A0-5 is a refined meta-model *i*. **Inconsistencies may happen in the event that some relevant objects or associations are lost when applying the evolution pattern, leading to narrowing of the scope of the meta-model. For example when applying the generalisation to introduce a generic service concept, one may lose a useful piece of information on specific service types. In this case, it is worth keeping service types as several concepts (object classes) in addition to the generic service concept.**

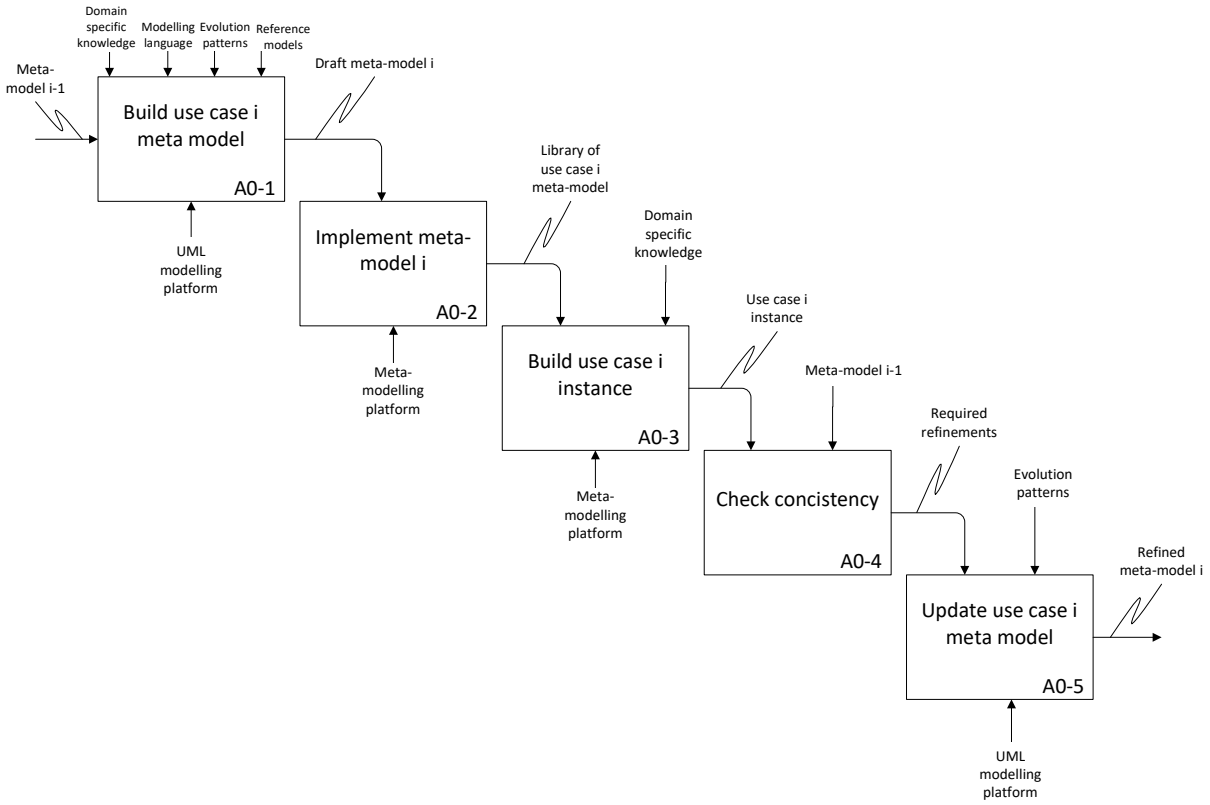


Figure 3: Implementation and iteration process details

4. Building a modelling and visualisation language for PSS engineering – case study

4.1. Overview and use cases

This section provides an overview of the application of the method and presents the three use cases which helped in developing a PSS modelling language. The development process spanned over one year involving the authors and other contributors. The development team includes domain-specific experts, researchers with background in model-based engineering and conceptual modelling, and software engineers.

The iterative process is jointly driven by three use cases linked to industrial research projects and by reference models from the literature. Each of the three research projects involves both industrials and academics including the authors' team. The three projects share a common general objective that is to design an industrial PSS-oriented offering. The three use cases from the three projects belong to different sectors and have different sizes and challenges. Consequently, they involve different complexity levels of the PSS engineering process.

Within the current case study, the aim of building a modelling language for PSS engineering is multi-fold: first, foster the reuse of the models representing PSS over many projects, second, contribute towards well informed decisions, third, ease the communication among the multi-disciplinary actors

involved in the engineering process (internally), and other stakeholders such as public institutions to promote the PSS potential (externally).

The real names of the use cases and the projects will not be disclosed for confidentiality purposes. All three cases involve manufacturing companies and are in the business-to-business context. In the following, the use cases are presented and the PSS solution is briefly mentioned. However, the details of the engineering process will not be reported on (Table 1).

Table 1. Overview of the use cases

	Use case 1 – C1	**Use case 2 – C2**	**Use case 3 – C3**
PSS function	**Water-saving sanitary equipment for healthcare facilities**	**Analysis of the physical properties of extracted stones for quarry production companies**	**Recycling steel sludge**
PSS content	**Shower head & maintenance services**	**Laser video system, with traceability and maintenance services**	**Compacting and briquetting equipment & maintenance services**
PSS provider	**1 SME**	**1 SME**	**1 equipment provider (A1)**
PSS customer	**Hospital**	**Stone quarry companies**	**Machining company (A2)**
Other actors			**PSS product customer (A3)**
Customer involvement	**Minor**	**Strong**	**Strong**

The first use case, C1, is a small SME (Small and Medium-sized Enterprise) providing water-saving sanitary equipment for healthcare facilities. The PSS project relies mainly on the efforts of C1. Other actors such as the customer (hospital) are only partly involved through a few interviews moderated by C1. The authors' team is also involved by defining services and potential PSS scenarios. C1 struggles to define an attractive offer for its main customer, a local hospital, involving a shower head and some cleaning and maintenance services. Such an innovative solution is coupled with a modular shower head easing the maintenance and spare part replacement. The envisioned PSS scenario involves C1 who will take care of all services for the hospital and a sub-contractor who will provide the shower head.

The second use case, C2, is an SME providing equipment to the quarry production market. The PSS project involves both the provider (C2) and the customer. The authors' team is also involved and supports usage analysis, service catalogue definition, and PSS scenario definition and assessment. The envisioned PSS solution consists of a conveyor integrating a laser video system to enable the analysis of the physical properties of the extracted stones. The two PSS scenarios envisioned consist namely of a sales-oriented and a use-oriented PSS. In each of the scenarios a set of services is provided which basically relates to the maintenance of the technical system and performance data recording. Unlike C1, where the customer role is minor, PSS relies here on strong customer involvement during both PSS design and operation.

The third use case, C3, consists of a consortium of several actors led by a research and development institution. The consortium involves an equipment manufacturing company (A1), a machining company (A2), and a smelting company (A3). The authors' team is involved as a sub-contractor supporting usage

analysis, service catalogue definition, and PSS scenario definition and assessment. The PSS project is driven by an innovative idea impacting the whole value chain of machining sludge. The idea consists of using a PSS-oriented offering to process the machining sludge generated by A2 using special equipment provided by A1. This would give rise to sellable briquettes made of the machining sludge which can be sold to A3. The PSS backbone is the briquetting equipment and maintenance services. Several scenarios were identified, each of which characterised by a different assignment of the roles to the actors of the value chain. However, the typical set of value creation activities is the same over the scenarios namely, manufacturing the equipment and using it to produce briquettes out of machining sludge. The equipment can be sold or rented to A2. The briquettes are sold to A3.

In all three cases, the work of the authors' team is closely related to the designers who focus on the tangible part of the PSS. A continuous interaction process took place in order to identify suitable services complementing product functions and increasing the value for both the customer and PSS provider.

The next two paragraphs illustrate the application of the method (§4.2) and elaborate on how reference models from the literature have fostered the development process (§4.3), respectively.

4.2.Iterative development process

This section reports on the general steps of the iterative process focusing specifically on the meta-model transformations brought by domain-specific knowledge.

PSS literature reveals a general agreement on some general requirements of the PSS engineering process. In particular, the PSS offering (including products and services), the actors involved and the infrastructure are put forth as a backbone for PSS design and development. These requirements provide a first set of guidelines to establish meta-model *0*. Use case C1 scenario is in line with these requirements and allows for some of them to be refined, in particular the infrastructure. Figure 4 shows the resulting meta-model *1* and an instance of use case 1.

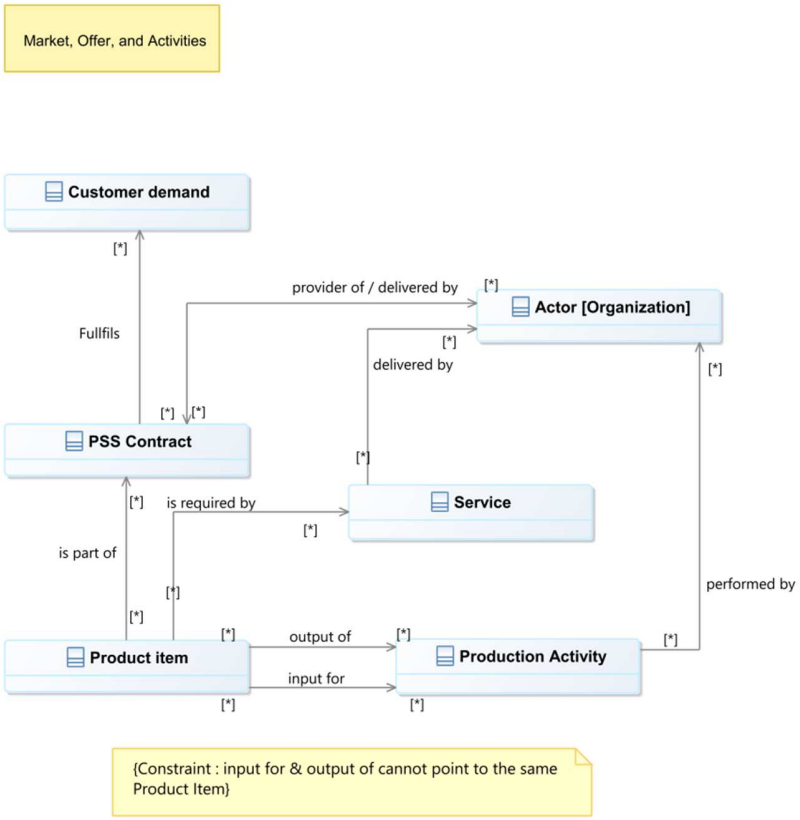


Figure 4: Use-case 1 derived meta-model

In the case of C1, there is no accurate insight into the scenario in terms of specific activities of each stakeholder of the PSS. This is partly due to lack of involvement of the various potential actors supporting the PSS. Furthermore, a clear vision of the performance indicators that might be used to evaluate the scenarios is missing within use case 1. The implementation of the meta-model in an ADOxx library is quite straightforward and no specific focus was put on graphical notations. The aim at this point is to provide a readable model.

Within use case 2, the offer includes both PSS and sales contracts, thus the contract concept was extended through a generalisation. PSS and sales contracts are inherited from a newly created parent class named contract. Furthermore, unlike C1, C2 is interested in a more established list of services with different types. Thus, a specialisation of services is introduced classifying them into training, maintenance and installation. This principle is consistent with the domain-specific knowledge as service packages are commonly used in both PSS and service industries. Similarly, product grouping can be used to put together product components or products sharing the same characteristics and which can be considered as one object. Basically, service and product groupings aimed at reinforcing use case instance readability.

C2's vision of the PSS scenarios is more advanced than C1, particularly due to the involvement of potential customers within the PSS project. Thus C2 is concerned with an assessment of the economic and operational performances of the scenario. This requires a set of economic indicators associated to contracts (to measure revenues) and activities (to measure costs), and operational performance indicators

related to the activities (e.g. inventory). In addition, specific data on the operators within the PSS scenario led to the introduction of an operator concept associated to the activities and services.

These extensions resulted in a more loaded model with additional concepts and associations. In order to keep the readability of the meta-model and subsequent instances, the meta-model was broken down into two views, namely demand and offer, and performance evaluation. These adaptations resulted in meta-model *2* which was implemented in an ADOxx library allowing to easily model PSS scenarios of both C2 and C1. The graphical notations are still basic at this point.

Use case 3 boosted the development of the modelling language further due to its wealth and PSS scenario maturity. As mentioned in the use cases overview, the PSS project involves multiple actors as the PSS spans over a major part of the machining sludge value chain. Several PSS scenarios were identified with different actor responsibilities. This required the introduction of a new concept in order to decouple activities from PSS actors so as to ease the configuration of the PSS scenarios. The newly added concept is named role, it links a set of activities to a given actor. Furthermore, the detailed information about the activities enabling a given service led to the extension of the activity concept through a generalisation. The activity became an abstract class from which inherit production and service activities.

While the implementation of meta-model *3* in an ADOxx library was straightforward, using this latter to model the PSS scenario of use case 3 did reveal some issues. In particular, the complexity of the model induced by the high number of interconnected concepts impedes the readability of the model. Consequently, meta-model *3* was broken down into 7 viewpoints, namely, demand, product, service, offer, organisation, activities, and scenario. All first six viewpoints provide the foundation for building the scenario viewpoint, in this sense they are used as pools of useful objects for scenario configuration. The graphical notations were also improved through identifying a colour protocol. Obviously, these adaptations are due not only to use case 3 but also the modelling experience acquired throughout use cases 1 and 2. The resulting meta-model *3* is shown in Figure 5.

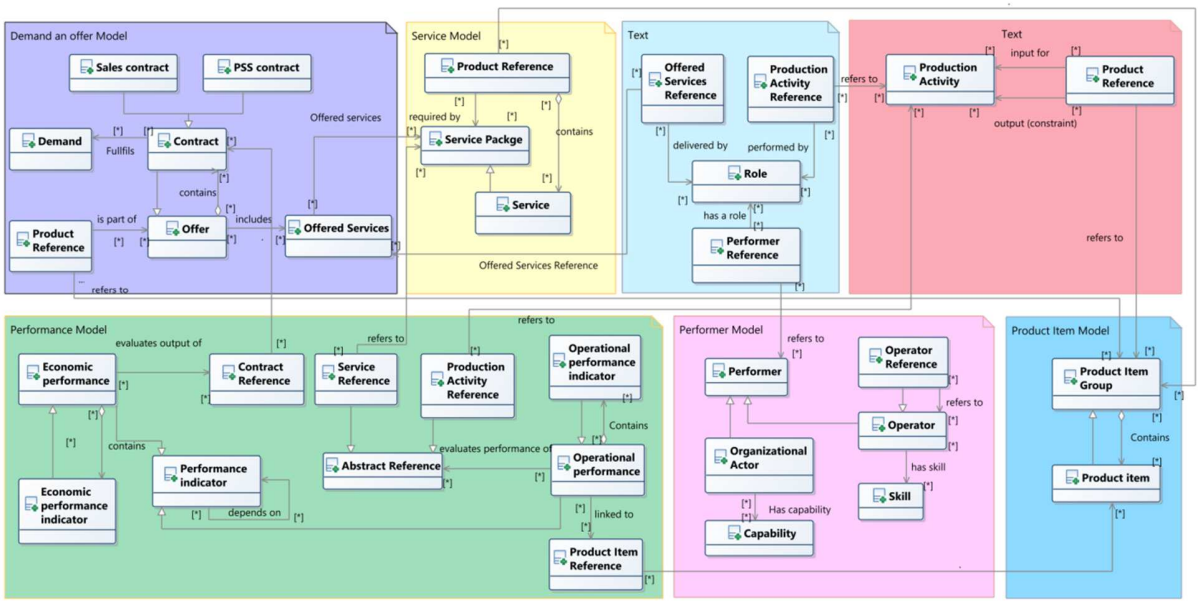


Figure 5: meta-model *3*

4.3. Contributions from the literature

The previous section illustrates how the PSS meta-model can be extended recursively through applying the method to use cases. By applying the same model transformation approach, the meta-model can be enriched with PSS modelling concepts from the scientific literature. This section shows a single iteration of the method involving the concepts from literature rather than case studies. In the area of PSS engineering and design, several state of the art papers emphasise various modelling methods (**Becker et al., 2010; Cavalieri & Pezzotta, 2012; Vasantha et al., 2012;** Qu et al., 2016). To achieve further exhaustiveness, **a review of the literature was pursued using the following keywords: e.g. PSS conceptual models, PSS generic models, PSS meta-model, PSS design support tools, PSS modelling method, PSS representation. This resulted in around 50 papers, mostly published before 2018 given the period when this step was carried out.** This set of contributions was further filtered in order to i) eliminate papers which do not provide enough conceptual insights on the models considered, and ii) avoid redundancy. This filtering resulted in a list of 10 papers, grouped below according to the modelling approaches discussed. Table 1 summarizes these results.

Table 2. Selected PSS modelling approaches

PSS Modelling Approach	*Description*	*References*
Extended Product Service Blueprint	Extensions of the "Service Blueprint" using BPMN (Business Process Modelling Notations) have been developed to take advantage of service-oriented concepts but also include product-oriented features to support service activities. This category refers to both 'Extented Service Blueprint' by (Hara et al., 2009) and 'Product-Service Blueprint' by (Geum and Park, 2011).	(Hara et al. 2009; Geum and Park, 2011)
PSS Layer Method	A multi-layer modelling framework which defines a meta-model of nine classes for a PSS. The framework provides a structured documentation to highlight requirements and tasks for PSS design.	(Müller et al., 2010)
Integrated Life Cycle	A modelling technique based on an integration of product lifecycle into service lifecycle. It also integrates a set of parameters to model cost, resource consumption and states of the product and the customer.	(Yang et al., 2010; Aurich et al., 2006)
Functional Hierarchy Modelling	A modelling technique that focuses on the representation of PSS functions. A novel PSS typology is suggested based on the level of integration between products and services and on the dominant PSS revenue mechanism.	(Van Ostaeyen, 2013)
Service Engineering	A multi-model framework for PSS design which allows the representation of critical concepts such as value, costs and functions of products or services. Both qualitative and quantitative models are used. They are distributed into 4 views: flow model, scenario model, scope model and view model.	(Sakao and Shimomura 2007; Sakao et al. 2009)
PSS Multi-View Modelling Framework	A multi-view modelling framework that combines both models used in product engineering and service engineering domains consistently with a systems engineering approach. The framework is focused on detailed design phases and supports interactions of PSS actors involved in the design process.	(Trevisan and Brissaud, 2016)
IPS² Metadata Model	A metadata reference model for PSS lifecycle management based on an analysis of the requirements of PSS providers and customers, and the different types of PSS.	(Abramovici et al., 2009)

The 10 key papers were analysed to extract useful pieces of knowledge and formalise this into several models represented using UML class diagrams. These models will support the extension of the PSS meta-model (Figure 5) through evolution patterns.

For illustration purposes Figure 6 shows a model derived from the Extended Product Service Blueprint. Basically both the extraction of the knowledge and the use of the resulting model to extend the PSS meta-model involve domain specific experts. For instance, the concepts 'Function' and 'Activity' show an improvement potential for the PSS meta-model. Function allows for describing PSS requirements regardless of the design solution, be it a product or service. This helps to broaden the design solution space so as to unleash the potential of product and service integration. Furthermore, with *Activity* being

limited to three types in the PSS meta-model, the derived model helps to cover further activity types. This is likely to help in clarifying PSS actor roles within value creation networks.

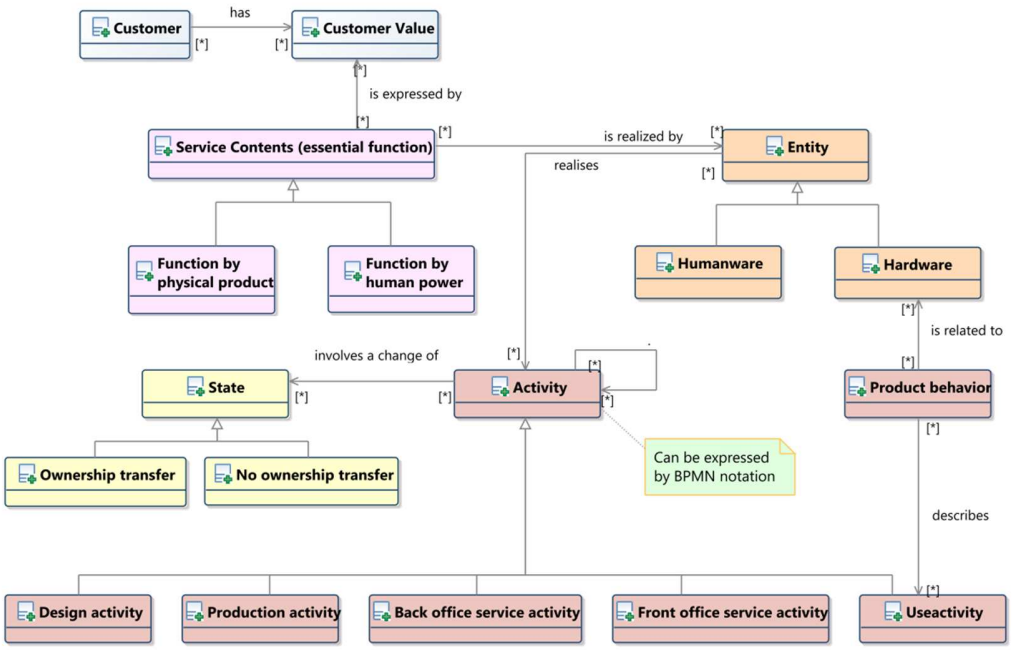


Figure 6: Extended Product Service Blueprint model

Following the same process as above, several additional concepts were identified and used to extend the PSS meta-model, some examples are provided in the following.

- *Periphery* represents a secondary product supporting the delivery of the PSS core products.
- *Value* specifies the expected benefits for the customers out of the PSS.
- *Use profile* characterises a dominant customer behaviour using PSS core product which could provide valuable information for defining services (customer autonomy, dependence, etc.).

The process of knowledge extraction from the literature led us to create two additional views and to consider changes in many concepts of the existing views. The final meta-model resulting from both the three industrial case studies and the integration of concepts from the PSS literature is presented in the next section.

5. A modelling language for PSS engineering

5.1. Meta-model overview

The PSS meta-model is structured into nine complementary modelling views (Figure 7). Each view comprises a set of specific concepts supporting a given area of the PSS design.

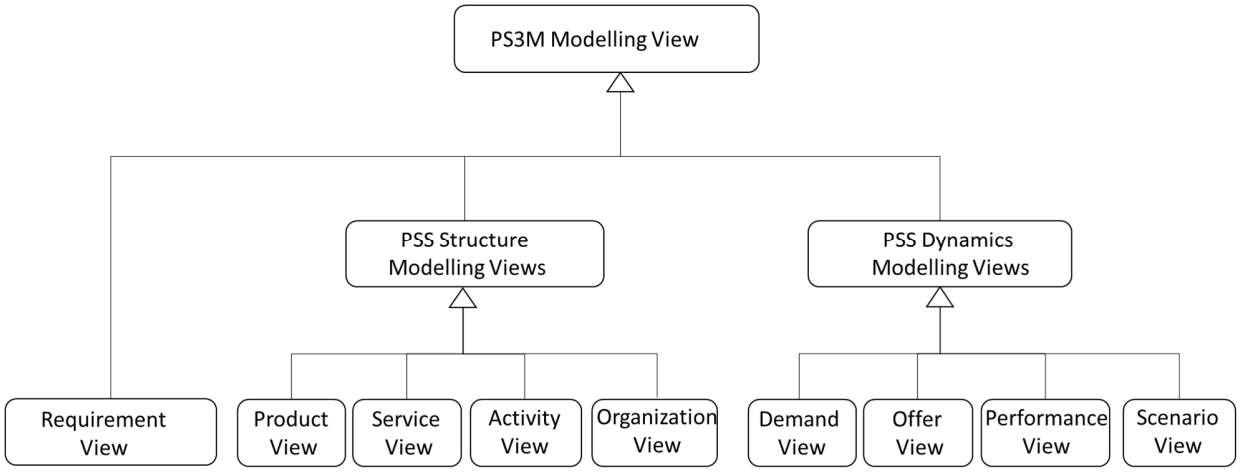


Figure 7: Modelling views

The *Requirement* view is aimed at capturing the end user value and functional expectations and translating them into a hierarchy of functions. The remaining views define the way to answer this function hierarchy. These views can be understood by referring to two main perspectives. The first perspective gathers four views dedicated to model the 'PSS Structure'. All the basic components required to define the architecture of the PSS offer and the associated delivery network are characterised through these four views:

- *Product* view is aimed at representing the overall structure of the core products, as well as the periphery products which support them, and to capture key technical features of these components resulting from the design decisions.
- *Service* view supports the progressive specification of services that can be delivered by the PSS provider throughout the PSS offer lifecycle (requirements, deployment, operation, retirement).
- *Activity* view represents the processes and activities required to build the PSS value chain. At this level, the value chain is still not configured, but all potential processes and activities are identified and characterised.
- *Organisation* view describes the required capabilities for PSS provision (collective capabilities of the firms) which are offered by the potential actors of the value chain. Two types of organisational actors characterise this view: internal and external actors (internal actors correspond to departments of the focal company of the PSS value chain). Any organisational actor can have one or more resources and can be characterised by its capability to take charge of a set of activities of value creation.

The second perspective gathers four additional views, dedicated to model the behavioural and dynamic features of the PSS offer. The final objective consists in modelling key dynamic aspects of alternative value chains configured to deliver the PSS offer. These characteristics are also linked to further design decision-making supports which are not within the scope of this paper.

- *Demand* view characterises the potential PSS markets and potential customer classes with specific use profiles.
- *Offer* view is used to specify the various alternative PSS offers which can be defined through combining objects from *Product*, *Service* and *Demand* views. Depending on the PSS offer content, alternative contracts can be defined specifying the economic model associated to the offer (use-, result- or performance-oriented contracts).

- *Performance* view defines sets of performance indicators to express PSS actor expectations from the PSS (economic, environmental, etc.).
- *Scenario* view is crucial in the design process as it allows for alternative PSS value chains to be specified by sharing the responsibilities among PSS actors. The scenario is defined by two main attributes: (1) a specific offer associated with a selling contract (detailed in the *Offer* view), and (2) an assignment of different roles to the PSS actors involved in the value chain depending on their capabilities.

All nine views are described by the meta-model in Figure 8.

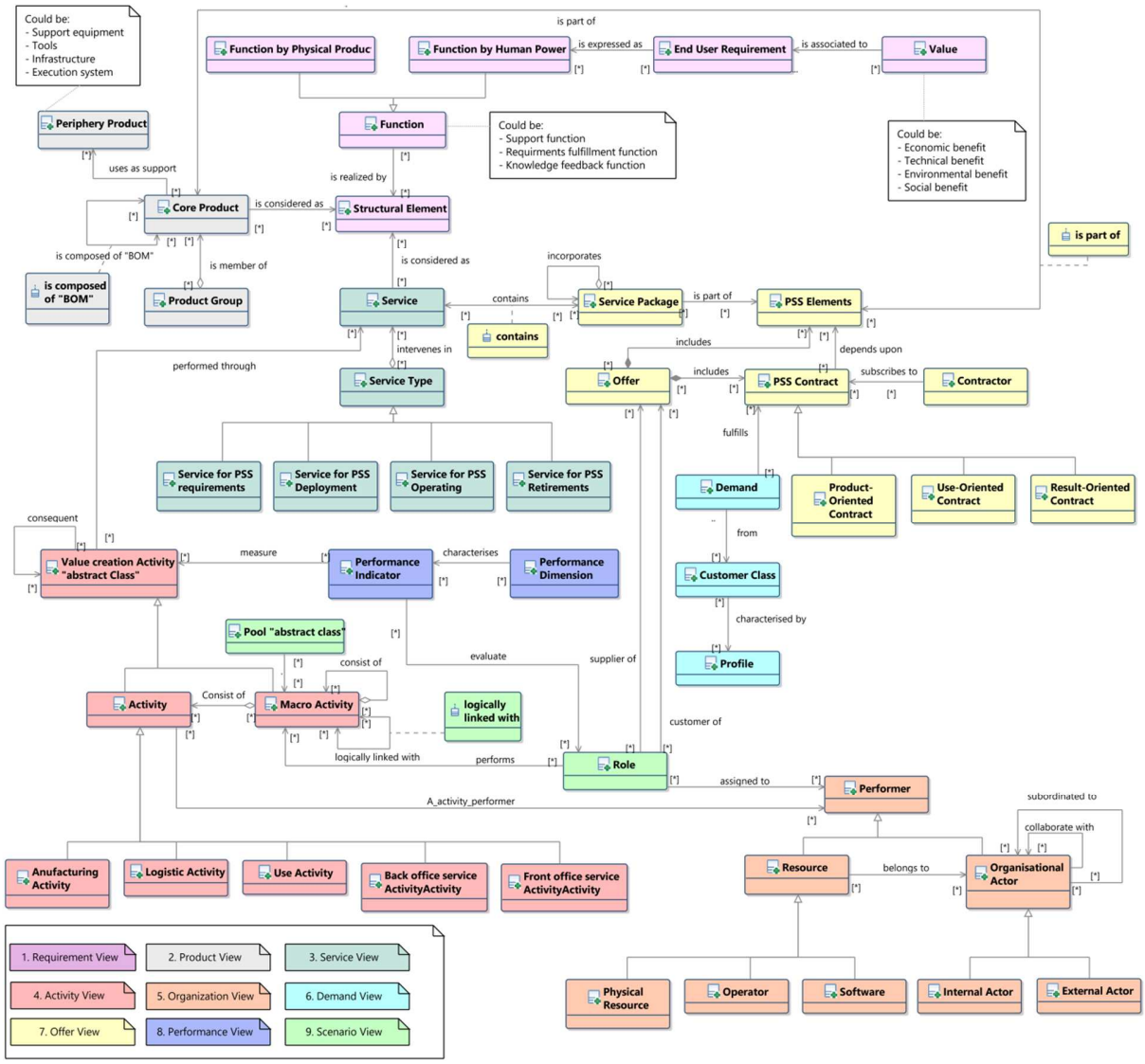


Figure 8: Resulting meta-model

5.2.Modelling tool: implementation and first experimentations

The modelling tool was developed iteratively as the proposed method suggests the implementation of each meta-model in an ADOxx library. These libraries are intermediary versions of the modelling tool. The resulting tool is named PS3M, which stands for Product-Service Systems Scenarios Modeler. The main graphical notations of the concepts used are summarised in Tables 2 and 3. The associations are represented with arrows or dashed arrows labelled with the association type. For each of the objects shown in Tables 2 and 3 there is a notebook including object attributes such as name, unit cost, etc. This information is useful for the subsequent steps of PSS engineering consisting in the economic assessment.

In order to ease the modelling process, a modelling procedure has been defined which organises the modelling viewpoints into a sequence starting from the requirements view up to the scenario view. Figure 9 shows a simplified view of the modelling procedure highlighting the main information flows between the modelling steps. The requirements view bridges customer needs and solution domain, providing an input for product and service view building. The offer as a combination between a contract and a set of products and services can only be determined by clearly defining product and service views. Afterwards, the activities required for product and service provision are defined. Subsequently, the organisation view is defined by specifying the capabilities of the PSS actors in terms of activities (who can do what). The performance view is then built consistently with the actors' required criteria for assessing the PSS scenarios. The demand view specifies information on the demand if any and can be defined at any time before defining the offer. The final step is to define the scenario based on previous views; more specifically a scenario is a given configuration of the actor network in terms of activities assigned to provision a given offer.

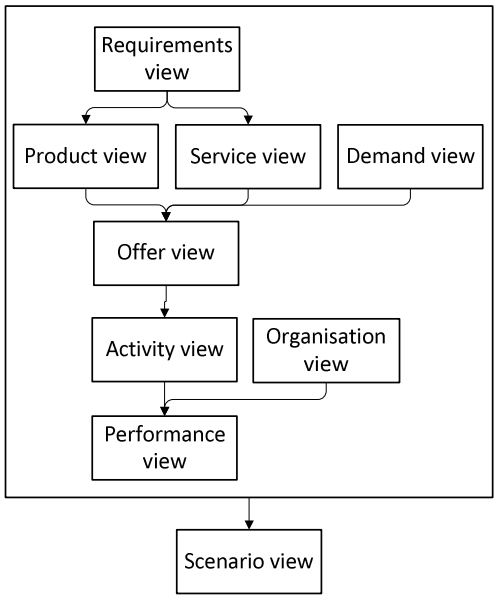


Figure 9: Modelling procedure

While the loops occurring during the modelling are not represented for readability purposes, these are of utmost importance. For instance there are usually loops between service and product views and between these views and the requirements view. The modelling tool is enhanced with a set of graphical notations aligned with the semantics of the concepts used. Some examples of these notations are highlighted in tables 2 and 3.

Table 3: Main graphical notations – PSS structure views

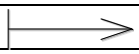
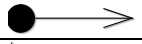
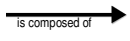
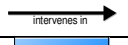
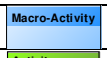
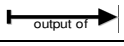
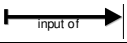
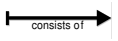
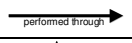
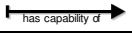
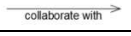
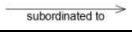
View	*Notation*	*Meaning – corresponding object in the meta-model*
Requirements view		Customer value
		End user (customer) requirement
		Function
		Technical function
		Service function – a function which can be fulfilled through a service
		Structural element – a product or a service
		Association linking an end user requirement to a customer value
		Association linking a customer value to an end user requirement
		Association linking a technical function to a structural element
		Association linking a service function to a structural element
Product		Product
		Association linking a product to its components
Service		Service group
		Service
		Association linking a Service to a Service group
Activity		Macro activity
		Activity (production, logistics, design, service)
		Product reference
		Service reference
		Association linking an Activity to its output Product
		Association linking an Activity to its input Product
		Association linking a Maco activity to its elementary Activity
		Association linking a Service to its realisatipon Activity
Organisation		Actor
		Activity reference
		Internal actor
		Operator (human resource)
		Software resource
		Physical resource
		Association linking an Actor to a potential Activity he/she can perform
		Association linking two Actors
		Association linking a resource to an Actor
		Association linking an Internal actor to an actor

Table 4: Main graphical notations – PSS dynamic views

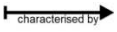
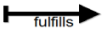
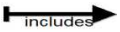
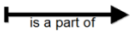
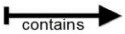
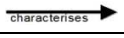
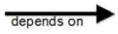
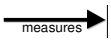
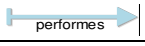
View	*Notation*	*Meaning – corresponding object in the meta-model*
Demand		Demand
		Customer class
		Customer profile
		Association linking a Customer class to a Customer profile
		Association linking a demand to a Customer Class
Offer		Offer
		Contract
		Product-Service System (PSS)
		Service package
		Association linking a Contract to a Demand
		Association linking a Contract to a PSS
		Association linking a Service Package to a PSS
		Association linking a Service to a Service Package
Performance		Performance dimension
		Performance indicator
		Macro-Activity reference
		Association linking a Performance indicator to a Performance dimension
		Association linking two Performance indicators
		Association linking a Performance indicator to a Macro Activity reference
Scenario		Actor reference
		Role
		Macro-Activity reference
		Indicator reference
		Offer reference
		decision node
		Association linking a Role to an Offer reference
		Association linking a Role to an Actor reference
		Association linking two roles
		Association linking a Role to an Offer reference
		Association linking a Role to a Macro-Activity reference
		Association linking a Performance indicator to a Role
		Association linking a Macro Activity reference to a decision node
		Association linking two Macro-Activity references

Figure 10 shows an extract of the PS3M tool interface highlighting the different model types corresponding to the views (on the left) and an example of a scenario view (on the right). The graphical notations were worked out iteratively in order to improve ergonomic aspects.

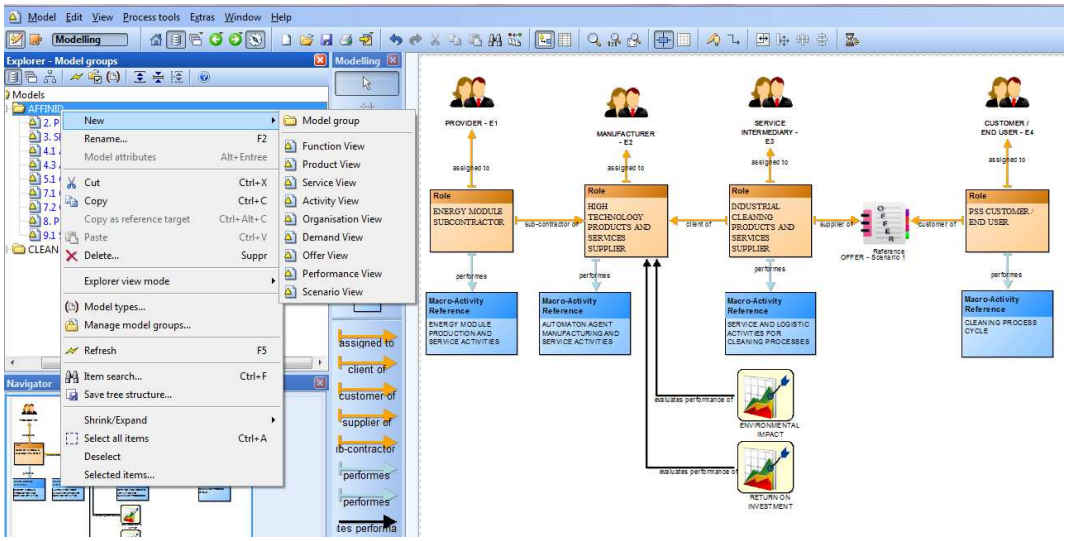


Figure 10: PS3M interface

The first experimentations of the modelling tool were launched to test its applicability to the PSS design process and its usefulness for practitioners. By applying the modelling procedure shown in figure 9, the tool users (designers, within a PSS design multi-disciplinary team) make the PSS progressively explicit and generate a knowledge repository covering the PSS design scope. The current version of the modelling tool, as a prototype, was used for two different industrial projects aiming at designing integrated offerings of products and services in a Business-to-Business context.

The first project is aimed at providing an industrial cleaning solution for the meat transformation industry. The PSS is based on an autonomous robotic cleaning solution and a set of potential associated services. This project was finished in June 2018 and took nearly 4 years. The second project is still ongoing and is focused on providing a PSS solution to improve quality control and traceability along the cheese production process. The PSS offer is based on a technological solution relying on RFID chips and a robotic system to put the chips on and take them off from the cheese.

The models built using PS3M supported both the understanding of the PSS, validation of various decisions throughout the engineering process and preparation of data collection for subsequent PSS quantitative analysis. These first two experimentations in a real industrial environment bear witness to the feasibility of using the PS3M tool and its underlying PSS design approach within industrial contexts. As a matter of fact, the cleaning solution developed within the first project and supported by the proposed modelling language, is at the final development stage right before market launch. Furthermore, both of the projects underline an acceptability of the modelling framework by practitioners as they were active users and contributors to the different modelling views.

Overall, the projects provide evidence of the usefulness of the modelling tool and help point out the areas of improvement which are related to ergonomic aspects. Further utilisation of the tool is likely to

uncover other opportunities for generalising the underlying meta-model and to provide further validation of its acceptability by PSS designers.

6. Discussion

The twofold contribution of the paper (proposed method and modelling language) helps to bridge two gaps in the literature. The first one relates to the need for a comprehensive modelling language (thus underlying meta-models) covering the various PSS dimensions involved in the engineering process. The resulting meta-model draws upon domain-specific knowledge derived from use cases and formal representation of the PSS as a system, using the conceptual modelling approach. This contributes to sharing the knowledge about the PSS so as to build a common understanding of the scenarios among cross-disciplinary teams. Additionally the alignment of the meta-model with the industrial requirements fosters the applicability of the modelling language.

The second gap addressed by the paper is due to the lack of guidelines for developing modelling languages for PSS. In fact, despite the amount of research addressing PSS modelling through ontologies, conceptual frameworks, etc., the methods for building such models are not made explicit. Thus the extension and even the use of these models are impeded in many cases. The iterative process underlying the method proposed within this paper shows how domain-specific knowledge and formal modelling approaches can be coupled to provide a modelling language with a targeted use.

The current version of the meta-model is quite comprehensive and has proven to be applicable to several use cases. However, some drawbacks still hinder full coverage of the PSS engineering scope. For example, the early steps of value expectations and value proposition definition are not addressed. This step goes hand in hand with the business model perspective which is only partially supported by the modelling language (Annarelli et al., 2017; Orellano et al., 2018). Besides, the current focus of the proposed approach is clearly only on the PSS design stage, whilst the operation and end-of-life stages are no less important. Therefore the extension of the modelling language presented to address PSS life-cycle is also an important perspective for future work (Hajimohammadi et al., 2017).

It can also be inferred from the application of the method to different use cases, that there is potential for it to be used in other emerging contexts where the main source of knowledge is the targeted domain itself, such as industry 4.0. **In this perspective, the research presented within the current paper promotes the use of models as a generic support for decision-making during systems engineering at large: one of the potential improvements of the current results consists in aligning the current meta-model with system engineering formal models**. This is consistent with recent advances in this domain which are vouched for by several initiatives such as the joint effort between the INCOSE and OMG for enhancing model-based system engineering **(Elgammal et al, 2017)**. Yet, the scope of the presented PSS meta-model and modelling language can be extended further through additional iterations of the development method.

The extension of the meta-model should however be addressed carefully in order to avoid high complexity. Trying to cover every single engineering situation may lead to overloading the meta-model with additional concepts, thus limiting the ease of modelling and usefulness of the modelling language. From a practical point of view, the unused concepts for the modelling are likely to hinder the use of the modelling language. In general, practitioners usually prefer simple and useful

tools rather than sophisticated methods hiding a lot of complexity in practice. Furthermore, during the extension of the meta-model, a set of hypotheses should be clearly delineated. This allows for sharing a common understanding of the goal of the meta-model and on what and what not to cover. Otherwise there is a risk of getting into an infinite loop of (irrelevant) updates of the meta-model, thus the modelling language.

Furthermore, although the modelling language was used in several projects from different sectors, specific skills were always needed to generate the models and the involvement of the PSS stakeholders was often limited to reading and discussing the models. This does not limit the relevance of the modelling output but rather points out some avenues of improvement. The collaborative development of the PSS could be reinforced if the modelling language became part of the language used by the PSS engineering team. In other words, the idea is to get to a point where there is no need to translate the natural language or CAD drawings into the proposed modelling language, but to simply use the latter to express one's raw ideas.

7. Conclusion

The current paper specifies a modelling language for PSS engineering using a step-wise process coupling domain-specific knowledge with formal modelling approaches. The usefulness of the method is evidenced through the development of the modelling language. The implementation and consistency check iterations provide proof of the concept of the modelling language and the underlying meta-model. The contribution of the paper is relevant not only to PSS engineering but also to model-based system engineering at large. Future research perspectives include a generalisation of the method to support systems engineering in the industry 4.0 context.

Both the case studies and the research projects which helped to develop the research presented in this paper, uncovered some areas of improvement in the modelling language. For instance, a major concern of the actors involved in PSS engineering relates to the economic assessment of the scenarios and to some extent, its environmental impact. Consequently, two areas to investigate are the extension of the meta-model to cover economic and environmental concerns from the early design stages and working the interoperability of the meta-model with other models for economic and environmental assessment.

Acknowledgements

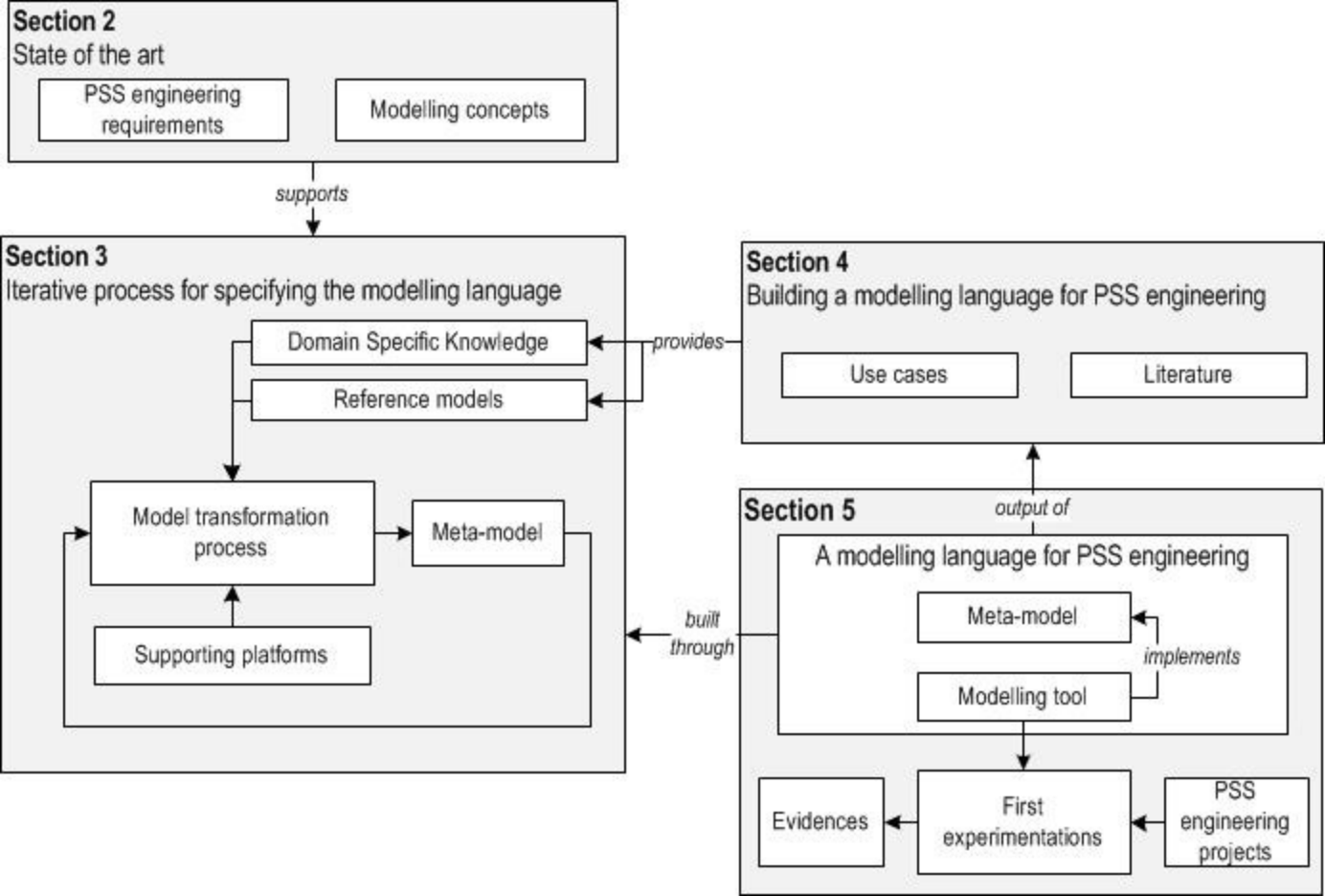
8. References

1. Abramovici, M., Neobach, M., Schulze, M., Spura, C., 2009. Metadata Reference Model for IPS2 Lifecycle Management. Proceedings of the 1st CIRP Industrial Product-Service Systems (IPS2) Conference, Cranfield University, 268–772.
2. **Annarelli A., Battistella C., Borgianni Y., Nonino F., 2017. Predicting the value of product Service-Systems for potential future implementers: results from multiple industrial case studies. Procedia CIRP 64 (2017), pp. 295-300.**
3. Aurich, J.C., Fuchs, C., Wagenknecht, C., 2006. Life cycle oriented design of technical product–service systems. Journal of Cleaner Production, 14(17), 1480–1494.

4. **Becker**, J., Beverungen, D.F., Knackstedt, R., 2010. Service systems: status-quo and perspectives for reference models and modeling languages, Information Systems and E-Business Management, 8(1) 33–66.
5. **Berkovich, M., Leimeister, J.M., Krcmar, H., 2011. Requirements Engineering for Product Service Systems. Business & Information Systems Engineering (BISE), 3(6), 369-380.**
6. Bertoni, A., Bertoni, M., Isaksson, O., 2013. Value visualization in Product Service Systems preliminary design. Journal of Cleaner Production, 53, 103-117.
7. Beuren, F. H., Ferreira, M.G., Cauchick, P.A., 2013. Product-service systems: a literature review on integrated products and services. Journal of Cleaner Production, 47, 222-231.
8. Bock, C., Dandashi, F., Friedenthal, S., Harrison, N., Jenkins, McGinnis, L., Sztipanovits, J., Uhrmacher, A., Weisel, E., Zhang, L., 2017. Conceptual Modeling, In: R. Fujimoto et al. (eds.), Research Challenges in Modeling and Simulation for Engineering Complex Systems, Simulation Foundations, Methods and Applications, DOI 10.1007/978-3-319-58544-4_3, Springer.
9. Boucher X., Brissaud D., Shimomura Y. (Edts), 2016. Design of sustainable Product Service Systems and their Value Creation Chains. CIRP Journal of Manufacturing Science and Technology, Special Issue, 15, 1-2.
10. Cavalieri, S., Pezzotta, G., 2012. Product-Service Systems Engineering: State of the art and research challenges. Computers in Industry, 63, 278 – 288.
11. **Creswell, J.W., 1994. Research Design: Qualitative & Quantitative Approaches, SAGE Publications, California.**
12. Durugbo, C., Tiwari, A., Alcock, J.R., 2011. A review of information flow diagrammatic models for product-service systems. International Journal of Advanced Manufacturing Technology, 52, (9–12), 1193–1208.
13. **Elgammal A., Papazoglou M., Krämer B., Constantinescu C., 2017. Design for Customization: A New Paradigm for Product-Service System Development. Procedia CIRP, 64, pp. 345-350.**
14. Frank, U., 2010. Outline of a method for designing domain-specific modelling languages, ICB-Research Report No. 42. Institute of Computer Science and Business Informatics, Essen, Germany.
15. **Geng, X., Chu, X., 2011. A New Pss Conceptual Design Approach Driven by User Task Model. Functional Thinking for Value Creation. Springer, pp. 123–128.**
16. Geum, Y., Park, Y., 2011. Designing the sustainable product-service integration: a product-service blueprint approach. Journal of Cleaner Production, 19 (14), 1601-1614.
17. **Hajimohammadi, A., Cavalcante, J., Gzara, L., 2017. Ontology for the PSS Lifecycle Management. In proceedings of the 2017 9th CIRP IPSS Conference: Circular Perspectives on product/Service-Systems, Copenhagen, Denmark. Procedia CIRP, 64, pp. 151-156.**
18. Hara T., Arai T., Shimomura Y., Sakao T., 2009. Service CAD system to integrate product and human activity for total value, CIRP Journal of Manufacturing Science and Technology, 1(4), 262–271.
19. Hribernik, K., Franke, M., Klein, P., Thoben, K.D., Coscia, E., 2017. Towards a platform for integrating product usage information into innovative product-service design. In proceedings of the 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Madeira Island, Portugal.
20. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., 2008. ATL: A model transformation tool. Science of Computer Programming, 72, 31 – 39.

21. Karagiannis, D., Buchmann, R.A., Burzynski, P., Reimer, U., Walch, M., 2016. Fundamental Conceptual Modeling Languages in OMiLAB. In : Karagiannis, D., Heinrich, M., Mylopoulos, C.J (Eds.), Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, Springer International Publishing, Switzerland.
22. Karagiannis, D., Heinrich, M., Mylopoulos, C.J (Eds.), 2016. Domain-Specific Conceptual Modeling, Concepts, Methods and Tools, Springer International Publishing, Switzerland.
23. **Khan, M., Wuest, T., 2018. Towards a methodology to design upgradable product service systems. Procedia CIRP, 78, 400-405. DOI: 10.1016/j.procir.2018.08.326.**
24. Klingner, S., Becker, M., 2012. Formal modelling of Components and Dependencies for Configuring Product-Service-Systems. Entreprise Modelling and Information Systems Architectures, 7(1), 44-66.
25. **Lee, S., Kim, Y., 2010. A product-service systems design method integrating service function and service activity and case studies. Procedia CIRP, 275–282.**
26. Lim, C-H., Kim, K-J., Hong, Y-S., Park, K., 2012. PSS Board: a structured tool for product service system process visualization, Journal of Cleaner Production, 37, 42-53.
27. **Maleki H., Belkadi F., Bernard A. 2018. A Meta-model for Product-Service System based on Systems Engineering approach. Procedia CIRP, 39-44.**
28. **Medini, K., Boucher, X., 2016. Configuration of Product-Service Systems value networks – Evidence from an innovative sector for sludge treatment. CIRP Journal of Manufacturing Science and Technology, 14-24.**
29. Marques, M., Poler, R., Agostinho, C., Jardim-Goncalves, R., 2017. An architecture to support the development of reconfigurable and updatable product-service systems in furniture sector. In Proceedings of the 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Madeira Island, Portugal.
30. Meier, H., Roy, R., Seliger, G., 2010. Industrial Product-Service Systems—IPS2. CIRP Annals - Manufacturing Technology, 607–627.
31. Müller P, Schulz F, Stark R., 2010. Guideline to elicit requirements on industrial product-service systems. In Proceedings of 2nd CIRP International Conference on Industrial Product/Service Systems, pp. 109-116.
32. Mylopoulos, J., 1992. Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development, New York: John Wiley and Sons.
33. National Institute of Standards and Technology, 1993. Announcing the Standard for Integration Definition for Function Modeling (IDEF0), National Institute of Standards and Technology, NIST.
34. Open Management Group, 2017, OMG® Unified Modeling Language® (OMG UML®) version 2.5.1, Open Management Group, Online : https://www.omg.org/spec/UML/About-UML/.
35. Qu, M., Yu, S., Chen, D., Chu, J., Tian, B., 2016. State-of-the-art of design, evaluation, and operation methodologies in product service systems. Computers in Industry, 77, 1-14.
36. Silva, A., R., 2015. Model-driven engineering: A survey supported by the unified conceptual model. Computer, languages, Systems and Structures, 43, 139 – 155.
37. Sakao, T., Shimomura, Y., 2007. Service engineering: a novel engineering discipline for producers to increase value combining service and product. Journal of Cleaner Production, 15 (6), 590-604.
38. Sakao T, Sundin E, Lindahl M, Shimomura Y (2009). A methodology for designing services modeling method, design method, CAD tool, and their industrial applications. In 'Introduction to Service Engineering', G. Salvendy and W. Karwowski, Edt John Whiley and Sons Inc., 2009.

39. Selic, B., 2003. The Pragmatics of Model-Driven Development, IEEE Software, 20(5), 19-25.
40. Silva, A.R., 2015. Model-driven engineering: A survey supported by the unified conceptual model. Computer languages, systems and structures, 43, 139 – 155.
41. Trevisan, L., Brissaud, D., 2016. Engineering models to support product-service system integrated design, CIRP Journal of Manufacturing Science and Technology, 15, 3-18.
42. Van Ostaeyen J., Van Horenbeek A., Pintelon L., Duflou J.R., 2013. A refined typology of product-service systems based on functional hierarchy modelling. Journal of Cleaner Production 51, 261 – 276.
43. Vasantha, G.V.A., Roy, R., Lelah, A., Brissaud, D., 2012. A review of product-service systems design methodologies. Journal of Engineering Design, 23 (9), 635-659.
44. Wellsandt, S., Terzi, S., Cerri, D., Norden, C., Ahlers, R., 2017. Model-Supported Lifecycle Analysis. In Proceedings of the 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Madeira Island, Portugal.
45. **Wiesner, S., Nilsson, S., Thobena, K.D., 2017. Integrating requirements engineering for different domains in system development – lessons learnt from industrial SME cases. Procedia CIRP, 351 – 356.**
46. **Wiesner, S., Freitag, M., Westphal, I., Thoben, K.-D., 2015. Interactions between service and product lifecycle management. Procedia CIRP 30, 36–41.**
47. **Wolfenstetter, T., Basirati, M.R., Böhm, M., Krcmar, H., 2018. Introducing TRAILS: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development. Journal of Systems and Software, 144, 342-355.**
48. Yang L., Xing K., Lee SH., 2010. Framework for PSS from service perspective. In Proceedings of the International Multi Conference of Engineers and Computer Scientists, IMECS, 1656–1661.
49. **Orellano, M., Lambey-Checchin, C., Medini, K., Neubert, G., 2018. The demand-pull approach to business model innovation through product-service systems: a case study, APMS 2018, Seoul, South Korea.**


Section 2
State of the art
PSS engineering requirements
Modelling concepts
supports
Section 3
Iterative process for specifying the modelling language
Domain Specific Knowledge
Reference models
Model transformation process
Meta-model
Supporting platforms
provides
Section 4
Building a modelling language for PSS engineering
Use cases
Literature
output of
Section 5
A modelling language for PSS engineering
Meta-model
Modelling tool
implements
built through
Evidences
First experimentations
PSS engineering projects