Technische Universität München

Fakultät für Informatik

Understanding Conflicts in Product-Service System Development

Mohammadreza Basirati, Master of Science

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Hans-Joachim Bungartz	
Prüfer der Dissertation:	1.	Prof. Dr. Helmut Krcmar
	2.	Prof. Dr. Manuel Wimmer

Die Dissertation wurde am 16.12.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13.09.2021 angenommen.

Preface

I am greatly thankful to my advisor Prof. Helmut Krcmar for the opportunities, which he provided and his guidance and support during my time at the chair. Furthermore, I would like to thank my research group leader, Dr. Markus Böhm, who kindly helped me through the challenges of my research and completing this dissertation.

I would also like to thank my colleagues at the chair, who were great companions along the way, especially, Jörg and Martin. Besides, I greatly benefited from professors and colleagues in research project SFB 768. Particularly, I would like to thank Prof. Vogel-Heuser and her chair, especially, Minjie Zou, who collaborated with me in the project, doing research, and writing papers.

During my Ph.D., I had the chance to supervise many students for their theses. This was a great experience. I became familiar with many interesting young minds and learned from the process of supervising them. Without them, this dissertation would have been very different.

Finally, I am deeply grateful to my family. First, to my love and my best friend, Morvarid, without whom I would have not been able to complete my journey in Ph.D. To my parents, who supported me and empowered me to begin such a journey. To my sister, Bita, and my brother, Ali, who were the first teachers I ever had.

Munich, December 2020 Mohammad R. Basirati

Abstract

Problem Statement: Globalized competitiveness on one side and growing environmental concerns on the other side have shifted more share of value towards services. Accordingly, the concept of product-service system (PSS) as an integrated bundle of products and services has emerged. At the same time, the fast advancements in software and digital technologies such as internet-of-things (IoT) accelerated such a shift. The software-side of any system has grown significantly providing new opportunities for service design and delivery. However, the software also increased the connectedness of a system's components, which are designed and developed by different teams from different domains. The resulting heterogeneity of components and individuals leads to prevalent conflict in PSS development. Nonetheless, PSS studies almost provide no insight into conflicting situations in PSS development. Besides, the studies on conflict management overlook the role of new technologies and the inter-connectedness of non-human elements.

Research Design: To fill this gap, we follow mixed-method research with the focus of providing practical knowledge and solutions. First, we conduct a structured literature review and qualitative expert interviews to analyze the circumstances in which IoT-integrated PSS is developed. Second, we conduct a survey and analyze it quantitatively using structural equation modeling. The survey investigates the relationship between different conflict types and project success. Moreover, we dive deeper into the reasons behind our findings from the survey through qualitative expert interviews. Finally, we follow a design science approach to develop methods and tools that facilitate conflict/inconsistency identification in PSS development.

Results: First, we provide a framework of IoT-PSS business model and a framework of IoT-PSS lifecycle management, which enlighten different aspects of PSS development at the time of advanced digital technologies such as IoT. Second, we analyze several hypotheses regarding different types of conflict's impact on project success. The findings confirm the negative correlation of conflicts with project success. Nevertheless, the impact of non-human-rooted conflicts (NHRC) is moderated by organization size and team size. NHRC is found to be negative for project success in large organizations and, interestingly, in small teams. The expert interviews revealed how dynamics of different settings (e.g. dependence on non-human elements) determine the impact of NHRC. In contrast to NHRC, the detrimental effects of human-rooted conflicts (HRC) remains almost unchanged in different sizes of organizations and teams. Third, we provide a holistic approach for systematic identification of conflicts/inconsistencies in PSS development, a tool that can assist such identification, and an ontology-based approach to encapsulate knowledge from various domains in PSS development to identify or avoid conflicts.

Contribution: The PSS-related findings and methods mainly contribute to PSS literature. They establish the foundations for more advanced conflict management in PSS development. Moreover, the proposed methods can be further applied, with or without alternation, to inconsistency identification in any system engineering field. Furthermore, the findings on conflict contribute significantly to the conflict literature. We contribute by introducing a new conflict classification (HRC vs. NHRC), providing empirical insight into the relationship

between conflict and project success, and showing the directions for future research. Both the PSS-related and conflict-related contributions are of high importance for practice. The proposed methods can be customized and realized in practice for better identification of conflicts in PSS development. Furthermore, the conflict-related findings have significant managerial implications. Practitioners can exploit such knowledge to better manage and resolve conflicting situations according to the settings (e.g. size of team or organization) and types of conflict (HRC vs. NHRC).

Limitations: The findings of interview studies highly rely on limited experiences and opinions of the subjects. We strived to overcome this weakness, first by using a structured literature review, and second, by a quantitative survey study. However, we do not claim that we were completely successful, particularly, as the other methods are limited in other ways. The conducted survey had a relatively low number of participants, which lowers the generalizability of its findings. Finally, the design science approach never results in the perfect solution, but iteratively searches for one. Hence, there is room for improvement in solutions proposed by this dissertation.

Future Research: This dissertation provides the foundations for an effective conflict management in PSS development. Our results can be further used and extended in several ways. First, future research can investigate the role of different conflict resolution strategies on project success and PSS development. The effectiveness of resolution strategies can be also analyzed with regard to the NHRC and HRC conflict types. Accordingly, future research should develop new methods and tools that support resolution of conflicts in PSS development. Furthermore, although we emphasized on importance of NHRC in PSS development, we have no empirical findings on how often different types of conflicts occur in practice and what are the most frequent causes of conflicts in PSS development. Such findings can support advancing the conflict identification methods of this dissertation.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

PSS	Product-Service Systems
IoT	Internet-of-Things
CPS	Cyber-physical System
HRC	Human-rooted Conflict
NHRC	Non-human-rooted Conflict
NLP	Natural Language Processing

Part A

1. Introduction

Products and services are becoming more and more integrated into bundles called product-service systems (PSS). With the increase in the number and variety of elements in PSS, the complexity of offerings and productions rises significantly. Heterogeneity among artifacts, individuals, tools, and processes makes conflict unavoidable and prevalent in PSS development. This dissertation investigates conflict in the context of PSS and provides methods and mechanisms to manage conflict in PSS development.

Motivation

Servitization has become an inseparable part of the business. Nowadays, most developed countries rely more on the share of the value created by services than physical products (Meier et al. 2010). Even manufacturing firms are evolving by integrating service provision into their core value drivers. More than 50% of manufacturers in the US and UK provide at least one type of service to the customers (Mastrogiacomo et al. 2019). The shift is very vivid for China with an increase from 1% servitized manufacturing firms in 2007 to 19% in 2011 and 38% in 2019 (Mastrogiacomo et al. 2019; Raja and Frandsen 2017).

Various factors contribute to the shift towards servitization such as increasing competitiveness of the global economy (Baines et al. 2007; Goedkoop et al. 1999; Meier et al. 2010; Peruzzini and Wiesner 2019; Vasantha et al. 2012), rising ecological restrictions (Komoto and Tomiyama 2009; Maussang et al. 2009; Meier et al. 2010), and higher demand for sustainable customer relationship (Meier et al. 2010; Vasantha et al. 2012). In this regard, the concept of product-service system (PSS) has emerged, which focuses on the intertwined development of products and services. A PSS aims at increasing profit by exploiting the synergy between product-side and service-side. The PSS concept is not necessarily equivalent to servitization (Meier et al. 2010; Mont 2002). Since the PSS perspective includes a service provider evolution by offering physical products besides service, while servitization merely focuses on enhancing physical products with service. Xerox provision of pay-per-print services instead of selling only print machines, BMW and Daimler joint car-sharing system called ShareNow, and Rolls-Royce "power by the hour" maintenance and support system for aero engines are distinct examples of PSS. Moreover, software and service companies such as Google and Facebook, now provide a variety of hardware parts. Google has become a successful smartphone manufacturer and Facebook produces smart video calling cameras.

Furthermore, significant progress in digital technologies and the emergence of new paradigms such as the internet of things (IoT) and cyber-physical system (CPS) have boosted the realization of PSS (Alexopoulos et al. 2018; Bressanelli et al. 2018; Seregni et al. 2016). Digitalization and ubiquitous software systems facilitate the connection between physical products and services (Bressanelli et al. 2018). The advancements of processors, sensors, and actuators speed up the service innovation and add more complexity to PSS. The integration of such advancements into products for providing services can be seen whether in simple products such as smart light bulbs or in complex products such as cars.

Thus, firms are experiencing a fast shift from traditional mere physical or mere service business models to intertwined offerings of a PSS at the same time of growth in the ubiquity of software.

The number and complexity of PSSs are increasing. Nowadays, a PSS consists of a higher number of elements than traditional physical products. To develop a PSS, a wide range of fields and expertise are required, which are more diverse than the past and at the same time, they need to collaborate frequently (Vasantha et al. 2012). Digital technologies have exacerbated such conditions by adding more volatility due to their requirements and their fast advancements (Pernstål et al. 2015; Song 2017; Wiesner et al. 2017).

In such circumstances, conflict emerges as a prevalent unavoidable phenomenon. Because the diversity of expertise within and between teams increases the number of misunderstandings and disagreements (Berkovich et al. 2011; Song and Sakao 2016; Wiesner et al. 2017). Every domain expert possesses a particular mindset and uses specific terminology, which may differ significantly from those of other domains' experts. Heterogeneity of functions among the team members is an acknowledged source of conflict, particularly in decision-making processes (Daspit et al. 2013; Levina 2005; Lovelace et al. 2001; Tsai and Hsu 2014).

Such heterogeneity is also reflected in non-human factors such as tools and methodologies. While a software development team is practicing the agile methodology, the physical product development teams may be using the traditional waterfall approach. Moreover, the components of products and services commonly are defined separately, while the components' requirements are highly interdependent. This characteristic of PSS development not only intensifies the misunderstandings but also increases the chance of inconsistency among the system's requirements (Song 2017). Even merely a higher heterogeneity in a system's requirements has been found directly associated with higher interpersonal conflicts (Liu et al. 2011).

Used interchangeably with notions such as disagreement, contradiction, or inconsistency, conflict is a broad and hard-to-define concept (Easterbrook et al. 1993). The Oxford English Dictionary defines conflict in such terms as "serious disagreement", "opposing feelings or needs", "serious incompatibility", and "being incompatible or at variance". Conflict is defined by Easterbrook et al. (1993) as the interaction of interdependent parties whose goals oppose and interfere with each other. They emphasize interaction, interdependence, and incompatibility to capture the meaning of conflict. Wall Jr and Callister (1995) define conflict as "a process in which one party perceives that its interests are being opposed or negatively affected by another party."

This dissertation introduces a new definition of conflict, which does not refute existing definitions, but rather allows more comprehensiveness. Because the existing definitions mostly point out to human nature of conflict and overlook the conflicting situations due to non-human elements. However, the incompatibility of tools, processes, and methodologies is very common in the development of PSS. This makes the existing definitions and studies of conflict incomplete.

According to the existing definitions, three major principles exist in any conflict. First, in a conflict, two or more elements are involved. Second, a relationship holds between the involved elements of a conflict. Third, the state of the elements and their relationships are different from the defined or expected objectives. Based on these principles, we define conflict as an "undesired variance between two or more related elements". Such a definition enables us to analyze the incompatibility and inconsistencies among non-human elements from a conflict lens.

Following the definition of this dissertation, conflict can be an indicator of project complexity, because the number of conflicts not only shows higher variance in the project, but also a higher number of elements and relationships. The extent of variance, the number of project elements and their relationships are considered as major indicators of project complexity (Geraldi et al. 2011). Furthermore, based on the new definition, a conflict may exist in both human and non-human aspects such as technical and organizational. Consequently, investigating conflicts can reflect the social and organizational complexity of a project as well as its technical complexity as the two main aspects of project complexity (Baccarini 1996).

Conflict is mostly recognized as a negative matter that impairs collaborations, behaviors, and structures (Wall Jr and Callister 1995). In cross-functional collaborations, such as PSS development, conflict can thwart innovation (Levina 2005; Tsai and Hsu 2014). Conflict often reduces the satisfaction of individuals (Jehn 1995; Jehn 1997) and the success of the groups (Guang-dong 2013; Jehn and Mannix 2001; Liang et al. 2010), particularly when the conflict tends to be more personal. Nonetheless, process conflict, the conflict between elements such as schedules and logistics, has been found also as a negative factor for team performance and project success (Greer and Jehn 2007; Guang-dong 2013; Jehn and Mannix 2001). In general, there is a strong relationship between project success and the work processes and methodologies (Joslin and Müller 2016; Khan et al. 2012). On a lower level, conflicts among requirements specifications are considered as one of the major reasons for the failure of software projects (Aldekhail et al. 2016). Moreover, several studies show the volatility and even variability of requirements artifacts are associated with reduced team performance (Shameem et al. 2018; Yang et al. 2015).

In summary, there is a huge growth in the intertwined development of products and services in frameworks such as PSS, which is facilitated by fast advancements in technology paradigms such as IoT. Such simultaneous and interconnected development of products and services has intensified the collaboration of experts from various domains, teams, and departments. Moreover, PSS is developed by various tools, methods, processes, and artifacts that are heterogeneous and commonly incompatible. Thus, such diversity in individuals and settings leads to a growth in number and variety of conflicts. Conflicts have a huge impact on how teams perform and projects evolve. Consequently, it is necessary to come to a better understanding of the conflict phenomenon that suits the new development settings of PSS. Accordingly, this dissertation focuses on reaching a deeper knowledge, methods, and mechanisms to deal with conflicts in PSS development.

A better understanding of conflict phenomenon points out circumstances, in which conflicts emerge during PSS development, the types of conflicts, and how the conflicts can influence PSS development. Consequently, practitioners can employ such knowledge to avoid or better manage conflicts. This directly improves team performance and decreases development time and cost. Furthermore, new conflict identification methods, which suit PSS development, can significantly assist practitioners in the efficient management of conflicts and moderating their negative impacts.

Research Questions

We aim to improve the understanding of conflict in PSS development and ways to identify conflicts in PSS. Particularly, we investigate the role of IoT and software as the main drivers of PSS. To this end, this dissertation follows three research questions (RQ):

RQ1: *What are the situations, from which conflicts in PSS development arise?*

This research question is the first step towards increasing our knowledge about conflicts in PSS development. Within this research question, we investigate how a PSS is conceptualized and realized, particularly using new technologies such as IoT. To this end, we conduct literature reviews and expert interviews. The results provide an overview of various types of PSS, how complex technologies enable the integration of products and services, and real PSS cases for more clarification. Hence, this research question provides the foundation for deeper analysis in the next research questions.

RQ2: *How do different types of conflicts affect project success?*

This research question analyzes different types of conflict and their impact on project success. Using survey, we investigate any correlation between presence of different types of conflict and the extent of project success. The results dive deeper into finding causes and mechanisms for such correlations using expert interviews. This research question increases our understanding of conflicts' influences on the project success.

RQ3: *How can conflicts be identified systematically in PSS development?*

The last research question strives to find solutions for the efficient identification of conflicts in PSS development. The results consist of tools, methods, and frameworks which facilitate different aspects of conflict identification and management. This research question completes our path for understanding conflicts in PSS development, starting from the context of conflict, to impacts of conflicts, and finally identifying and managing conflicts.

The first and second research questions generally increase our understanding of conflicts in PSS development. The answers to these two questions tell us about various types of conflicts, situations from which they arise, and how they influence the project success. The third research question mostly focuses on how to identify or manage the conflicts in PSS. Thus, the research questions cover various aspects and complete our view on conflict in PSS development.

Structure

This dissertation consists of three parts. Part A starts with the problem definition and research questions. Afterward, we address the conceptual background of this dissertation, that is the PSS-related concepts such as PSS types and PSS development processes, and conflict-related concepts. Besides, we provide a preliminary case study, which indicates various conflicting elements of conflict in PSS development. As the next step, we describe the research approach of this dissertation. The first and second research questions are mostly based on expert interviews and surveys for understanding the conflict and the third research question is mostly based on the design science approach. Part B1 consists of two peer-reviewed publications which are the main publications of this dissertation. Part B2 provides extra four related publications, not included for evaluation of the dissertation. The first and second publications address PSS conceptualization and realization through IoT. The third publication empirically investigates the impact of conflict on the project success. The rest of the papers provide either a framework, method, or tool for the identification and management of conflicts in PSS development. Finally, in Part C, we provide a summary of the results presented in Part B1 and B2 (main papers and other related papers), a discussion of the results, their implications, a`nd open issues for future research. The summarized structure of this dissertation is presented in Figure 1.

Part A		Introduction, Conceptual Background, Sources of Conflicts, Research Approach
Part B1		Main Publications (For Evaluation)
RQ2	P3	**Understanding Relationship of Conflict and Success in Information System Development** *Method: Mixed-method of Quantitative Survey and Qualitative Expert Interviews*
RQ3	P4	**Towards Systematic Inconsistency Identification for Product-Service Systems** *Method: Design Science*
Part B2		Other Papers (Not Included for Evaluation)
RQ1	P1	**IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation** *Method: Literature Review and Expert Interviews*
RQ1	P2	**Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation** *Method: Literature Review, Expert Interviews, and Case Survey*
RQ3	P5	**Introducing TRAILS: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development** *Method: Design Science*
RQ3	P6	**Facilitating Consistency of Business Model and Technical Models in Product-Service Systems Development: An Ontology Approach** *Method: Design Science*
Part C		Summary of Results, Discussion (Implications and Limitations), Future Research

Figure 1 -Detailed structure of the dissertation

Part B1 presents main embedded publications of this dissertation which should be reviewed and evaluated. We provide short descriptions of P3 and P4 as the main papers (ordering is based on the research questions) in the following.

P3: Understanding the Relationship of Conflict and Success in Software Development Projects (Basirati et al. 2020) – This paper empirically investigates whether conflict influences the success of a software-intensive project and if yes, how and through which dynamics. Types of conflict introduced in the Conceptual Background of this dissertation are used for the analysis of this paper. To this end, we used a mixed-method research approach. First, we conducted a survey and quantitatively analyzed it using structural equation modeling (SEM). Second, we conducted multiple expert interviews to further understand how the identified relationships between conflict and project success work.

P4: Towards Systematic Inconsistency Identification for Product-Service Systems (Basirati et al. 2018) – Inconsistencies among tools, processes, and artifacts are prevalent in PSS development. This paper, first, clarifies various types of logical inconsistencies that can emerge in PSS development. Second, it provides a framework and guidelines through which inconsistency identification can be practiced. Third, the paper elaborates on the approach in a PSS scenario. The results of this paper can be adapted and implemented in industrial PSS use cases.

Part B2 presents other papers published and related to this dissertation (they are not part of any evaluation for this dissertation). In the following, we present short descriptions of these papers.

P1: IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation (Basirati et al. 2019a) – This paper investigates the combination of IoT and PSS from two aspects: business development and system development. It sheds light on the current practices and situations, in which PSS is developed. Hence, it clarifies in which circumstances conflicts arise in PSS development. To this end, we employed a systematic literature review and multiple expert interviews. The results have two folds. First, a framework of PSS-IoT business models. Second, a framework of IoT for PSS lifecycle management.

P2: Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation (Basirati et al. 2019b) - This paper is a continuation of the first paper. Hence, it tackles exactly the same research gap, which is how a PSS is defined and developed through IoT. This paper completes P1 by providing real industrial cases for each described situation in the paper. Using the real cases, the paper clarifies various aspects of PSS development in the presence of complex technologies such as IoT.

P5: Introducing TRAILS: A Tool Supporting Traceability, Integration and Visualisation of Engineering Knowledge for Product-Service Systems Development (Wolfenstetter et al. 2018) – This paper presents a tool that enables the integration and visualization of various models during PSS development. It assists practitioners by clarifying the links among the heterogeneous models in PSS development.

P6: Facilitating Consistency of Business Model and Technical Models in Product-Service Systems Development: An Ontology Approach (Zou et al. 2019) – One of the major challenges in PSS design is to align the design of the services initiated by the business model with the development of physical products. Particularly, it is challenging to trace the dependencies between the services and product parts. This paper tackles this challenge by providing an ontology approach that integrates parameters from both the business and service side and the physical side. Through this approach, practitioners can map how a change in the business model can affect the design of coupled services and products.

Table 1 - List of embedded publications according to RQ order

No.	Authors	Title	Outlet	Type
P1	Basirati, Weking, Hermes, Böhm, Krcmar	IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation	PACIS 2019	Conf.
P2	Basirati, Weking, Hermes, Böhm, Krcmar	Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation	Asia Pacific Journal of Information Systems	Journal
P3*	Basirati, Otasevic, Rajavi, Böhm, Krcmar	Understanding the Relationship of Conflict and Success in Software Development Projects	Information and Software Technology	Journal
P4*	Basirati, Zou, Bauer, Kattner, Reinhart, Lindemann, Böhm, Krcmar, Vogel-Heuser	Towards Systematic Inconsistency Identification for Product-Service Systems	International Design Conference 2018	Conf.
P5	Wolfenstetter, Basirati, Böhm, Krcmar	Introducing TRAILS: A Tool Supporting Traceability, Integration and Visualisation of Engineering Knowledge for Product-Service Systems Development	Journal of Systems and Software	Journal
P6	Zou, Basirati, Bauer, Kattner, Reinhart, Lindemann, Böhm, Krcmar, Vogel-Heuser	Facilitating Consistency of Business Model and Technical Models in Product-Service Systems Development: An Ontology Approach	9th IFAC Conference 2019	Conf.
* P3 and P4 are the main two publications of the dissertation for evaluation, other papers are complementary for covering the concepts and are not part of any evaluation or review for the dissertation.				

In addition to these publications, we conducted and were involved in several additional studies, which are related to the third research question. The list of additional papers (AP) is presented in Table 2. AP1 presents a tool that automatically identifies interdependencies among requirements specifications of a system, which are written in natural language. The study follows a design science approach. The tool uses machine learning and natural language processing (NLP) techniques to build a model that classifies pairs of requirements as interdependent or non-interdependent. The knowledge of interdependencies among requirements assist practitioners, particularly product owners and managers, to prioritize requirements more efficiently and avoid conflicts among requirements and the teams. Because conflicts occur where there is an interdependency. Besides, interdependency between the two requirements reflects interdependency between the individuals and teams who are assigned to implement those requirements. Such knowledge is of high importance in PSS development since the relationship between teams, products and services, and different components of the system is not clear. Therefore, the tool of AP1 supports the identification and management of conflicts among requirements as well as the individuals and teams. AP2 introduces a new method for alignment between documents of requirements and engineering models. Hence, it facilitates keeping consistency among information stored in different artifacts. Consequently, the new method reduces the probability of conflict. AP3 develops a new inconsistency management approach based on the defined framework of P4. Moreover, it presents a simple implementation of the approach. The paper focuses mostly on conflicts among models and potential escalation of such conflicts among teams and departments.

Table 2 - Additional publications related to third research question

ID	Authors	Title	Outlet	Type
AP1	Basirati, Böhm, Krcmar	REINALYZE: A Machine-Learning Tool for Identifying Requirements Interdependencies *(Submitted)*	Requirements Engineering Journal	Journal
AP2	Koltun, Basirati, Hammeed, Böhm, Krcmar, Vogel-Heuser	Reverse Engineering on changed Functional Specification Documents for Model-Based Requirements Engineering *(Published)*	2019 International Conference on Industrial CPS	Conf.
AP3	Kattner, Bauer, Basirati, Zou, Reinhart, Lindemann, Böhm, Krcmar, Vogel-Heuser	Inconsistency Management in Heterogeneous Models - An Approach for the Identification of Model Dependencies and Potential Inconsistencies *(Published)*	International Conference on Engineering Design 2019	Conf.

2. Conceptual Background

In this section, we explain the fundamental concepts of the dissertation. First, we discuss various aspects of the PSS concept. Subsequently, we explain the concept of conflict and its types. Finally, we dive deep into constituting elements of conflicts from a socio-technical system (STS) theory lens. Accordingly, we introduce human-rooted conflict (HRC) and non-human-rooted conflict (NHRC) as two new types of conflict.

Product-Service System

Since the first introduction to product-service systems (PSS) in 1999 by Goedkoop, Van Halen, Te Riele, and Rommens many new definitions have been proposed in the literature. Table 3 presents various definitions of PSS chronologically. All the definitions share in common that a PSS aims to increase effectiveness. Most of the definitions mention that such an increase is achieved by improving the relationship with the customer through providing services. Decreasing dependence on the environment and raw material is another major motive for a PSS. The latest definition by Meier et al. (2010) highlights the software-intensive and knowledge-intensive nature of PSS.

PSS is considered as a powerful source of competitive advantage and sustainability (Ardolino et al. 2016). PSS can increase the profit margins, provide new growth opportunities in saturated markets, and build long-term customer relationships. Overall, PSS benefits the PSS provider, customers and consumers, the environment, and the society (Beuren et al. 2013).

Table 3 - Definitions of PSS adapted from Annarelli et al. (2016)

Definition	Reference
"A product–service system is a marketable set of products and services capable of jointly fulfilling a user's need. A product is a tangible commodity, manufactured to be sold. A service is an activity (work), often done on a commercial basis and for others, with an economic value. A system is a combination of elements including their relations."	(Goedkoop et al. 1999)
"A business innovation strategy offering a marketable mix of products and services jointly capable of fulfilling clients' needs and/or wants - with higher added value and a smaller environmental impact as compared to an existing system or product."	(Manzini et al. 2001)
"A system of products, services, supporting networks and infrastructure that is designed to be: competitive, satisfy customer needs and have a lower environmental impact than traditional business models."	(Mont 2002)
"A system consisting of tangible products and intangible services designed and combined so that they jointly are capable of fulfilling specific customer needs."	(Tukker 2004)

"Products and services which can simultaneously fulfil people's needs and considerably reduce the use of materials and energy."	(Halme et al. 2006)
"A social construction, based on "attraction forces" (such as goals, expected results and problem-solving criteria) which catalyse the participation of several partners. A PSS is a result of a value co-production process within such a partnership. Its effectiveness is based on a shared vision of possible and desirable scenarios."	(Morelli 2006)
"A market proposition that extends the traditional functionality of a product by incorporating additional services."	(Baines et al. 2007)
"An attempt to use existing industrial and commercial structures to create radically environmentally improved products by treating them as services."	(Evans et al. 2007)
"Industrial PSS can be defined as a systematic package in which intangible services are attached to tangible products to finish various industrial activities in the whole product life-cycle."	(Jiang and Fu 2009)
"An Industrial Product-Service System is characterized by the integrated and mutually determined planning, development, provision and use of product and service shares including its immanent software components in Business-to-Business applications and represents a knowledge-intensive sociotechnical system"	(Meier et al. 2010)

Three types of PSS have been introduced and accepted as the main PSS types in the research: product-oriented, use-oriented, and result-oriented PSS (Baines et al. 2007; Tukker 2004; Yang et al. 2009). Table 4 describes the three types of PSS in terms of their underlying business-model elements (Reim et al. 2015).

Another way of looking at the three types of PSS is to consider what point they have reached on the innovation scale; result-oriented PSS is the most innovative, and product-oriented PSS is the least innovative. For a PSS to evolve from product-oriented to result-oriented, it may take incremental steps and/or a radical path. Incremental innovation, in this context, means that product-oriented PSS evolves slowly to use-oriented and then further to result-oriented. This happens through a slow and steady continuous-improvement process. Radical innovation, on the other hand, means that product-oriented PSS transforms directly into result-oriented PSS, skipping the use-oriented stage. This often involves a radical shift in technology and leads to a total reconfiguration of the PSS.

The Xerox case, mentioned in the previous section, is a typical product-oriented PSS example. All manufacturers that provide maintenance and recycling services besides their products can be considered examples of the product-oriented PSS type.

Car-sharing and bike-sharing cases belong to use-oriented PSS type. In such cases, the price is calculated based on the units of usage. For example, BMW car-sharing service DriveNow and Daimler car-sharing service car2go (joint together under ShareNow) charge the users based on a per-minute basis. The users can take any available car in the city and park it for free anywhere in the city. The cars (physical products) are typical car models manufactured by BMW and Daimler. However, these car manufacturers do not sell their cars in the PSS, but they use their physical products as a means to deliver mobility services to the users.

Result-oriented PSS has the highest level of servitization, in which the service-side creates more share of value than the product side (Yang et al. 2010). If a washing machine manufacturer provides its machines for free and charges the users on a pay-per-use basis, this would be a use-oriented PSS. It is possible to incorporate more servitization in such a PSS by delivering laundered clothes, i.e. the result, instead of the machines (Baines et al. 2007). Such a system would be a result-oriented PSS. A real-case advanced example of result-oriented PSS is Lufthansa's AVIATAR digital power platform, which provides various apps and services for airlines and their suppliers and partners. For instance, the airlines can create networks with each other and share their airplanes' spare parts with the purpose of increasing the availability.

Table 4 - Different types of PSS according to Tukker (2004) and Reim et al. (2015)

	Product-oriented	Use-oriented	Result-oriented
Value creation	The main responsibility of provider is service delivery.	The main responsibility of provider is the usability of the product or service.	Results are the main responsibility of provider.
Value delivery	Provider delivers extra services in addition to sold products (e.g., maintenance or recycling).	Provider focuses on service usability along with product usability.	The results are counted as the main deliverables instead of products or services.
Value capturing	Customer pays for the product and extra delivered services.	The payment is performed over usage phase continuously (e.g., leasing).	Customer pays based on outcome units instead of pay-per-use or pay-per-product.

Conflict

Conflict term covers various situations and aspects that are also reflected by other notions such as disagreement or inconsistency. For example, requirements engineering literature analyzes conflict and inconsistency in requirements without differentiating them substantially, either in written specifications (Gervasi and Zowghi 2005; Liu 2016) or generally between stakeholders' perspectives (Sommerville and Sawyer 1997). Nonetheless, there are more references to conflict, when there is a discourse about disagreement among individuals. Different perspectives and understandings of conflict led to various definitions of it. Table 5 lists the definitions of conflict and inconsistency.

Table 5 - Definitions of conflict and inconsistency

Terminology	Definition	Source
Inconsistency	"any situation in which two parts of a specification do not obey some relationship that should hold between them".	(Easterbrook and Nuseibeh 1996)
Inconsistency	"a state in which, two or more overlapping elements of different software models make assertions about the aspects of the system they describe which are not jointly satisfiable"	(Spanoudakis and Zisman 2001)
Conflict	"the interaction of interdependent people who perceive opposition of goals, aims, and values, and who see the other party as potentially interfering with the realization of these goals	(Putnam and Poole 1987) and (Easterbrook et al. 1993)
Conflict	"a process in which one party perceives that its interests are being opposed or negatively affected by another party."	(Wall Jr and Callister 1995)
Conflict	1) "A serious disagreement or argument, typically a protracted one. a. a prolonged armed struggle b. a state of mind in which a person experiences a clash of opposing feelings or needs. c. a serious incompatibility between two or more opinions, principles, or interests."	Oxford Dictionary
Conflict	"Undesired variance between two or more related elements"	(Basirati et al. 2020)

The conflict emerges and acts as in three high-level parts: causes of conflict, core conflicting situation, and effects of conflict. The effects of conflict can in a cyclic manner become causes of conflict. This is depicted in Figure 2. In Section 0 of the introduction, we elaborate on the first part, which is the causes of conflict. The first research question of the dissertation addresses partly the core conflict situation. The second research question investigates the third part that is the effects of conflict, particularly regarding the project success. Finally, the third research question goes one step further than the level of conflict nature and analyzes methods and tools for the identification of conflict.

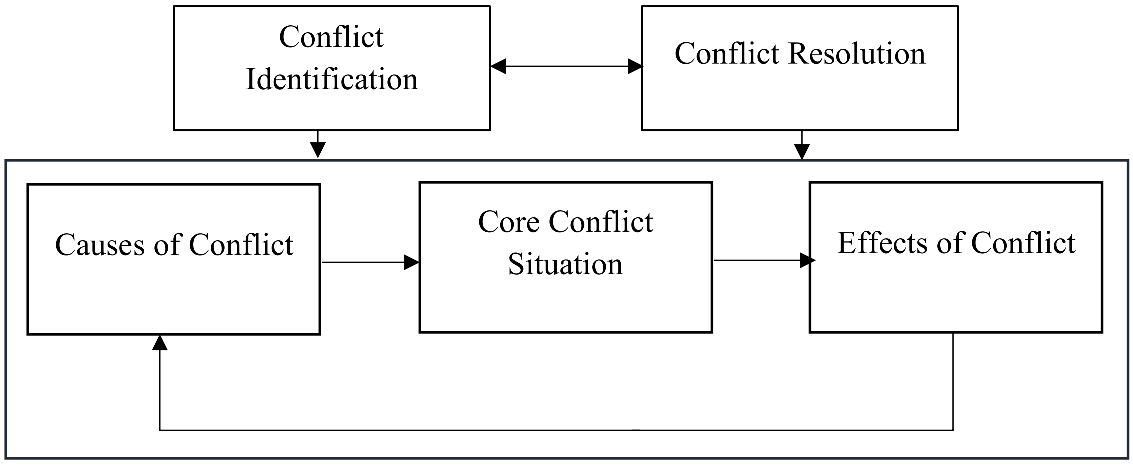


Figure 2 - Cyclic nature of conflict adapted from Wall Jr and Callister (1995)

Conflict may emerge in different levels. For example, a conflict may occur even within an individual, when one's role or duty is in contrast to the person's values or skills. Similarly, a conflict may emerge between two persons (interpersonal conflict), two teams (inter-team conflict), and two departments (interdepartmental conflict). We also emphasize the non-human aspect of conflict by considering elements such as structure, process, and artifacts as another level of conflict.

To study conflict, the literature has introduced various types. Because conflict is inherently a multidimensional concept that can be studied and classified from various aspects. We can distinguish among conflicts by their causes (e.g. disagreement, competition, lack of trust, etc.), the constituting elements of conflict (e.g. conflict of interests, conflict of role and skill, etc.), the types of individuals, among whom conflicts emerge (e.g. superior vs. subordinate, minority vs. majority, etc.), the level and granularity level of conflict (e.g. interpersonal conflict, interdepartmental conflict, etc.), and in general, all the factors that can be associated with the three parts depicted in Figure 2.

Relationship conflict, task conflict, process conflict, and role conflict are relevant types of conflict that can be associated with this dissertation. Relationship conflict is concerned with incompatibility in attitude and style of individuals that usually leads to tension and animosity (Jehn and Mannix 2001; Simons and Peterson 2000). For example, the differences in political preferences, values, and personal characteristics are relationship conflicts (De Dreu and Weingart 2003). Task conflict is about disagreements and differences in viewpoints regarding the tasks of a group (Jehn and Mannix 2001; Simons and Peterson 2000). For instance, disagreement about goal definitions of a task is task conflict (Greer et al. 2008). Process conflict deals with incompatibilities about task accomplishment aspects such as resource allocation (Jehn and Mannix 2001). Tensions in scheduling, logistics, and coordination of individuals are examples of process conflict (Behfar et al. 2011). Role conflict emerges from the inconsistent demands of a single or multiple job role(s) (Ghorpade et al. 2011). For example, a role might be incompatible with an individual's capabilities, values, resources, or even other roles (Rizzo et al. 1970).

The above-mentioned four types of conflict are amongst the most studied types of conflict in the literature. Such types of conflict hardly cover the influence of non-human elements in conflicts. Only process conflict partly incorporates the incompatibilities between two non-human element such as schedules. However, the importance of non-human elements such as technological innovations (e.g. in PSS) is increasing and we are more dependent on technology than the past.

To tackle this issue, we take a socio-technical system (STS) theory lens in this dissertation. Next section discusses a preliminary study that identifies the conflicting elements, which constitutes conflicts and classifying conflicts into two high-level types of human-rooted and non-human-rooted conflicts (HRC and NHRC). Table 6 shows the relation between HRC and NHRC, and above-discussed types of conflict.

Table 6 - Relation between existing conflict types and HRC and NHRC

	HRC	NHRC
Relationship Conflict	●	○
Task Conflict	●	◐
Process Conflict	◐	◐
Role Conflict	●	○

Conflicting Elements

In this section we present the results of a preliminary study that is conducted as part of this dissertation. The study employs expert interviews, literature review, and rather a small case study to construct a taxonomy of conflicting elements (see Figure 3). The taxonomy clarifies what elements constitutes a conflict from a socio-technical system (STS) theory. According to STS, a work system consists of two jointly independent, but correlatively interacting systems of social and technical. Based on STS theory, both the social and the technical systems create the output jointly (Bostrom and Heinen 1977). Thus, STS enables us to reach a comprehensive view on PSS development, which is affected highly by human, i.e. social system, and non-human factors, i.e. technical system. The resulting taxonomy of this section acknowledges the distinction between conflicts caused by human factors and non-human factors. Such a distinction clarifies the role of non-human factors in creating conflicts.

Every element in the taxonomy may be part of a conflict (depicted in Figure 3). The elements are grouped into five dimensions: interest, background, activity, work setting and artifact. Interest, background and activity dimensions constitute human-rooted elements. Work-setting dimensions are non-human-rooted elements, in addition to the artifact dimension, which presents the facts, logics, technologies and standards that exist in PSS development. We define every element briefly and present a summary of the findings from the interviews and the literature review.

Human-rooted	***Interest***	Goal				Priority		
	Background	Domain Knowledge	Skill	Personality	Org. Culture	Regional Culture	Natural Language	Terminology
	Activity	Role				Task		
Non-human-rooted	***Work Settings***	Tool	Approach	Org. Structure	Spatial	Monetary	Temporal	
	Artifact	Content				Format		

Figure 3 - Taxonomy of conflicting elements (conflict sources)

Interest: Design and management of a systems' requirements inherently incorporates diverse stakeholders, and it is certainly true that each stakeholder aims for a particular goal. In addition, the importance and urgency of the objectives are different from stakeholder to stakeholder. We reflect these two aspects in goals and priorities. According to the interviews, requirements evolve due to implementation considerations expressed by designers and developers. For instance, such evolution based on the priorities of the developers was partly in conflict with the goals and priorities of the customers, which had been expressed in the first place. In another case, business-side stakeholders added requirements, regarding which they had no idea how they could affect system performance. Since the architect was aware of those affects, he had different priorities, which seemed absurd. Furthermore, an interviewee observed that during the maintenance phase, there was great turnover in stakeholders, and new people came along with new ideas, goals and preferences, which were in conflict with the old ones. Another situation experienced was when collaborators were presenting the other parties and the information was transformed with some modifications. Consequently, the communicated goals and priorities were in conflict with the actual ones. Likewise, one study states that a representative group of a whole might not be able to reflect the views of all related employees in every organization, which causes conflicting biased and incomplete perspectives (Bhat et al. 2006). A special example of this challenge is in large IS projects, which involve several departments from each organization, although only a few departments represent the whole (Holmström and Sawyer 2011).

Background: An influencing aspect in any system development is the background of each stakeholder. Background elements can uncover hidden agendas and help to understand and anticipate others' behaviors and habits. *Organizational Culture* stands for behavioral norms and habits that every working unit creates, whether it is a team, a department, or an organization. A reported experience is that a person from another organization confirmed accomplishment of a project milestone, when in fact it had not occurred. Based on the interviewee's arguments, the conflict arose because of how differently two organizations treat schedules and agree on what is done. In literature, several studies emphasized on this conflicting element and argued that conflicting organizational cultures have an even more significant influence in comparison to natural cultures (Berenbach 2006; Chakraborty et al. 2015). *Domain Knowledge* stands for the particular deep knowledge that a collaborator has regarding a specific topic. Several experts have stated that it is challenging to avoid misunderstandings when people from different domains are involved, simply because much deep knowledge and many details are missing from each other's perspective. One situation experienced was for every collaborator to add new details to the problem definition without considering the other domains' constraints, which leads to conflicts. The literature shows a trend towards more cross-functional collaboration, as the systems are becoming more complex. Many studies discussed the conflicts emerging from collaboration of people with different backgrounds, which mostly leads to misunderstandings (Bjarnason and Sharp 2015; Damian and Zowghi 2003). For example, Damian and Zowghi (2003) observed that people from different domains deal with different conflicting abstraction levels. It is also noteworthy to mention that one study argues that conflicts among people with dissimilar backgrounds can be leveraged to support the learning process (Coughlan and Macredie 2002), which is in conformance with general studies on interpersonal conflicts (Wall Jr and Callister 1995). The *Skill* element reflects the collaborator's broad knowledge and abilities aside from the domain knowledge, such as expertise in a specific framework (e.g. Scrum) or tool (e.g. Jira). For example, introducing "agile" in traditional industries led to problematic situations. *Personality* stands for characteristics of an individual, his/her general style, preferences and social behaviors. It has been observed frequently by the experts that people simply do not communicate with each other because they do not like each other's personalities. However, two experts argued that a team needs to include people with different personalities that support and complement each other in order to be successful. Personalities can even alter our behavior in using a tool. A study showed that conflict between a personality and a tool can decrease performance and satisfaction (Aranda et al. 2010). *Regional Culture* determines specific norms, habits and characteristics that a collaborator holds, which has its roots in the culture he or she is from. As the interviewees were working on many international projects, they noticed numerous differences between their own culture (mostly European) and the cultures of other regions (mostly Asian and South American). In conformance with the expert interviews, several studies also show that the people from different countries have conflicting cultures with regard to how to deal with organizational hierarchies (Bhat et al. 2006; Hanisch and Corbitt 2007). *Natural Language* presents how differences in the various collaborators' level of expertise in the language that they are using can lead to conflict. Several experts experienced misunderstandings and barriers to communication due to natural language differences. Several interviewees mentioned this type of conflict as an important barrier. In the literature, development and evolution of requirements is even considered as a language development process, which the quality of the language highly influence on the quality and success of the project (Rosenkranz et al. 2013). *Terminology* stands for specific terms and expressions that are used and have a meaning in a particular context and work environment. Based on the interviews, conflicts caused by terminology differences were experienced in collaborations involving diverse domains or work environments, particularly whilst collaborating with an offshore organization. It is also reported frequently in literature that different departments develop different terminologies, which can be conflicting with each other (Azadegan et al. 2013; Bjarnason and Sharp 2015; Daneva et al. 2014).

Activity: PSS development incorporates individuals who are active in a wide range of roles such as business analyst, product manager, developer, technician, tester, and so on. For every role, a set of tasks is defined. A common experience is that the number of roles and the separation of duties led to conflicts, because introducing a high number of roles adds new role requests, and results in a higher probability of their collision. Moreover, a separation of roles causes a separation of knowledge, which leads to the need for more communication, and consequently to more delays. Another situation observed by an expert was that a role did not fit to the methodology (agile) of the project. Furthermore, in one case, an expert reported an experience in which a role's tasks were in conflict with those defined at the outset, and which were expected. One interviewee found that it is very common that people violate the defined scope of their role responsibilities, by performing out-of-scope tasks. Hence, they do tasks which conflict not only with their defined roles, but also with the others' tasks.

Work Settings: PSS development activities take place in particular environments and settings. While interests, background and activity present human-rooted elements of a conflict, work settings address most of the non-human-rooted elements of a conflict. Work Settings elements reflect how a collaboration is carried out, in what organizational structure, and with how much time and money. *Tool* demonstrates the aspects related to using tools during an RDC, whether they be collaboration-related tools (e.g. Skype or Wikis) or general tools. Many interviewees experienced conflicts between non-similar tools, which disrupted the development. In particular, if the collaborators are working remotely, we saw more complaints about how dissimilar tools blocked the communication. As reported by experts and literature, conflicting tools decrease communication significantly. This leads to information loss and more new conflicting situations. *Approach* stands for the methods, processes and methodologies that an individual or organizational unit employs in order to develop and accomplish the goals of the project. One expert argued that it is vital to know the process, otherwise the best tools would be useless if they cannot match to the process. Another expert experienced that in a project, new employees had no idea about the existing processes. Consequently, each person started to have his/her own interpretation and made a new change to the process, which led to many conflicts. Similar situations were reported in literature, particularly when a physical product manufacturer and a new software department collaborate, which is common in PSS development: the latter use agile methodology in contrast with the traditional waterfall model of the manufacturer (Pernstål et al. 2015). *Organizational Structure* demonstrates the structural and hierarchical characteristics of teams, departments and organizations. One expert argued that people exploit organizational structures in order to avoid their tasks. Another expert observed that people from a higher level of the hierarchy had no knowledge about the details of what was going on, leading to some conflicts. The same finding has also been reported in a study (Bjarnason and Sharp 2015). Some structures have been found conflicting with the processes and methodologies, for example, decision-making loop structures that in a case study led to incomplete deliverables (Bhat et al. 2006). Some of the main reported structural conflicts are power differences among collaborators, which highly influence their interactions (Bjarnason and Sharp 2015; Macaulay 1999). *Spatial* stands for spatial characteristics of work settings, such as whether it is spatially distributed or collocated or whether a collaboration is taking place in a single large hall or in a small room. Moreover, spatial differences such as working from two different sites can be counted as a conflict if it is undesired and worsens the collaboration.

As many experts experienced, distributed settings can intensify most of existing challenges and lead to a higher number of conflicts. Such experiences include conflicts in culture, language, tools and so on, which are similar to reports from the relevant literature (Damian and Zowghi 2003). *Temporal* represents time-related aspects of a collaboration, such as schedule, duration and time pressure. Many experts working in global projects experienced a time zone conflict. Such a conflict is not only between two time zones, but also – as one expert said – it conflicts with how you work particularly if you work using an agile methodology. Based on the literature, the emergent nature of requirements and their high volatility has led to significant number of emergent collaborations (Inayat and Salim 2015; Marczak and Damian 2011), which provides more conflicts of schedules and plans. Two studies reported conflicting schedules as a barrier as well as a consequence of frequent collaboration (Bjarnason and Sharp 2015; Pernstål et al. 2015). *Monetary* represents budget-related aspects of a collaboration, in particular how budget policies and budget limits restrict or enable certain actions in the development. One expert observed that in an inter-organizational collaboration, differences in budgeting by the organizations led to conflict. Moreover, an observed situation is that the budgeting policies of an organization were based on fixed contracts and completely defined requirements, which was in conflict with their approach, i.e., agile.

Artifact: The other non-human-rooted elements that can lead to conflicts are artifacts and the information that they hold. During PSS development, a high number of artifacts are developed, and may also be received as inputs. These artifacts hold a significant amount of knowledge and logic of the final system. Consequently, the content of these artifacts may be inconsistent, such as inconsistent requirement statements in documents (Gervasi and Zowghi 2005). Furthermore, the format of these artifacts can be conflicting, as can the standards and guidelines which form the basis for the development of the artifacts. Several experts mentioned that artifacts were built in an overly domain-specific manner that is hard to understand for other teams. In one case, an expert experienced that conflict between two parties' ways of structuring the documents wasted a considerable amount of time. Moreover, artifacts and information can be formulated in different data types and technologies, or can be conformed to different standards; cross-disciplinary collaborations in particular lead to such conflicts (Pernstål et al. 2015).

According to the introduced taxonomy, we define human-rooted conflict (HRC) and non-human-rooted conflict (NHRC). HRC consists of conflicting elements that are categorized under human-rooted elements. In contrast, NHRC consists of non-human-rooted elements of the taxonomy. We formally define HRC as a conflict that is rooted essentially in human factors, which are related to the general interests or background of a person such as personality or culture. In contrast, NHRC is a conflict that is exclusively rooted in non-human factors such as tools, processes or artifacts (Basirati et al. 2020). Table 7 provides several concrete examples of HRC and NHRC.

Table 7 - Concrete examples of HRC and NHRC

HRC	NHRC
A conflict between stakeholders' priorities	A conflict between DevOps' tools and the real needs of the DevOps process
A conflict between two teams from different organizations (e.g. in an outsourcing scenario) caused by different organizational cultures	A conflict between tools used by different teams from different organizations or departments
A conflict between two teams from different departments caused by a difference in terminologies	A conflict between methodologies of two teams such as agile and waterfall
A conflict caused by challenging or incompatible personalities	A conflict between two work processes e.g. by blocking each other
A conflict between individuals due to different level of expertise	Inconsistencies between legacy code and the new code libraries

3. Research Approach

This dissertation follows a pragmatic approach, which does not limit the research by emphasizing on the problem to be researched and its implications (Tashakkori et al. 1998; Yvonne Feilzer 2010). Within this approach, the research iteratively moves back and forth between induction and deduction (Morgan 2007). Moreover, a pragmatic approach transfers knowledge between different types of research methods to clarify best the answers to the research question and their implications (Morgan 2007).

Through the pragmatism paradigm, we follow mixed-methods research, which employs various qualitative and quantitative research methods to find the best answer for the research question. Teddlie and Tashakkori (2010) discuss several characteristics for mixed-methods research, which we explain some of them. In mixed-methods research, the researcher selects and synergistically integrates the most suitable qualitative and quantitative methods to thoroughly analyze a phenomenon of interest. More importantly, mixed-methods research advocates diversity at all levels of the research. Such diversity is not reflected only in the research methods, but also in types of research questions (e.g. confirmatory and exploratory), data sources, conclusions, and argumentations. In other words, in mixed-methods research, the researcher uses triangulation on many levels to reach the best possible solutions and results. However, in contrast to classic triangulation that strives to converge the results, divergent results are also accepted in mixed-methods research. Another aspect of mixed-methods research is its iterative, cyclic approach, which rotates around the observations, theories, inferences, and hypotheses. Finally, the emphasis in mixed-methods research is highly on the research problem that allows employing the best suited set of research methods.

Ability of mixed-methods research to address both confirmatory and exploratory research questions is of great value. For example, in the second research question of this dissertation (see Section 1.3, Figure 1), we first investigate whether there is any correlation between different types of conflict and project success using quantitative analysis. Subsequently, we explore the reasons for such correlation using expert interviews. Furthermore, the integrating interferences from various research methods can offset the limitations of a single method without weakening their strengths. In addition, different methods may lead to divergent (contradictory) findings that complete our perspective and understanding on the phenomenon of the interest. Such abilities of high value as they better clarify different aspects of the research problem and the mechanism through which we made the conclusions.

This dissertation follows mixed-methods research to understand the phenomenon of conflict in PSS development and to provide solutions for the identification of conflict. The main purpose of employing this approach is to reach a complete overview of the conflict in PSS development, whether in which circumstances conflicts emerge and how they affect or how conflicts can be identified. The qualitative side of this dissertation is mainly based on expert interviews. Moreover, we use survey and structural equation modeling (SEM) to quantitatively analyze conflicts. Finally, following a design science approach, we apply the knowledge collected from different sources to design and develop new methods and tools for the identification of conflicts.

Overview of Research Methods

In this section, we present the research methods used in the dissertation and the publications of it (see Table 8). The first research question (P1 and P2) was answered mostly by structured literature review and expert interview. We also did a survey of industrial cases in first research question (exactly in P2), however as it was not conducted systematically, we did not include it in Table 8. To answer the second research question (P3), we performed a survey, a structural equation modeling (SEM), and expert interviews. Finally, we followed a design science approach to answer the third research question (P4 - P7).

Table 8 - List of research methods used in the publications

ID	Publication	Lit. Review	Expert Interview	Survey	SEM	Design Science
P1	IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation	X	X			
P2	Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation	X	X			
P3	Understanding the Relationship of Conflict and Success in Software Development Projects		X	X	X	
P4	Towards Systematic Inconsistency Identification for Product-Service Systems					X
P5	Introducing TRAILS: A Tool Supporting Traceability, Integration and Visualisation of Engineering Knowledge for Product-Service Systems Development					X
P6	Facilitating Consistency of Business Model and Technical Models in Product-Service Systems Development: An Ontology Approach					X
P7	Inconsistency Management in Heterogeneous Models - An Approach for the Identification of Model Dependencies and Potential Inconsistencies					X

Literature Review

Literature review plays a critical role in conducting any research, since they establish the foundations for acquiring and cumulating, and advancing knowledge (Brocke et al. 2009; Webster and Watson 2002). Rowe (2014) defines literature review as a research method that "synthesizes past knowledge on a topic or domain of interest, identifies important biases and knowledge gaps in the literature, and proposes corresponding future research directions." Literature reviews on a mature topic can deliver us with a conceptual model and a thorough understanding of the topic. Regarding emerging topics, a literature review is able to expose the theoretical foundation that suits the topic (Webster and Watson 2002).

Nonetheless, a literature review may have various purposes such as summarizing, integrating, or criticizing prior results, theories, research methods, and applications of theories (Brocke et al. 2009). From a slightly different perspective, Kitchenham et al. (2009) name three major reasons for conducting a systematic literature review, which are summarizing the existing findings, identifying research gaps, and to create a research framework. Paré et al. (2015) classify literature reviews into more detail types that consist of narrative reviews, descriptive reviews, scoping/mapping reviews, meta-analyses, qualitative systematic reviews, umbrella reviews, theoretical reviews, realist reviews, and critical reviews.

Webster and Watson (2002) consider the literature review as a concept-centric method. According to such a perspective, the concepts are units of analysis and may belong to different levels of abstraction or dimensions. For example, conflict causes and conflict consequences build one level of abstraction, and concepts of organizational conflict versus individual conflict belong to another level of abstraction. This concept-centric consideration is opposed to an author-centric literature review that merely presents a summary of the publications from a set of authors.

Brocke et al. (2009) introduce a circular framework for conducting literature reviews that consists of five major phases: definition of review scope, the conceptualization of topic, literature search, literature analysis and synthesis, and research agenda. Kitchenham et al. (2009) provide thorough guidelines on how to perform a literature review systematically. The guidelines consist of three major phases: planning the review, conducting the review, and reporting the review. Each phase has several steps. Planning the review starts with the identification of the need for a review, specifying the research questions, and establishing a review protocol. The conducting phase includes the identification of research and search queries, quality assessment, data extraction, and data synthesis.

According to Kitchenham et al. (2009) and Brocke et al. (2009), a literature review is as follows. It starts with a research question. Accordingly, we identify relevant research streams, relevant journals, outlets, and research databases. As the next step, we formulate the search query. To this end, iteratively we test various keywords and their combinations to find the search query with the most relevant and most comprehensive results. Subsequently, we collect and select studies that are commonly performed in several rounds. It starts by reading the title and abstract of the papers and further going deeper into the complete text of the papers to investigate whether a paper is relevant or not. Moreover, backward and forward searches can help us to reach a more complete set of relevant studies. The next step is to assess the quality of the studies. This can be done based on different inclusion and exclusion criteria that are dependent on the field of the research. Finally, the most important step is to collect data from the selected studies and synthesize new findings. To this end, the concept-centric approach of Webster and Watson (2002) is of high importance.

Expert Interview

Expert interview is a valuable and common tool to collect data about a phenomenon. Interview is a distinct method as it directly engages experienced subjects to provide deep contextual information from their observations, experiences, and interpretations (Schultze and Avital 2011). Interviewees can express their perspective from a world that is usually not observable and accessible by other individuals (Schultze and Avital 2011).

According to Myers and Newman (2007) there are several major types of expert interviews: structured, unstructured, semi-structured, and group interview. A structured interview has a complete script of what actions should be done and what questions should be answered. Hence, there is no room for improvisation in structured interviews. In semi-structured and unstructured interviews, some questions or topics are prepared to be addressed, however, the discussion might expose new particular questions. In a group interview, which can be structured or unstructured, the researcher perform interview with more than one interviewee at the same time.

The script of an expert interview must at least include the following sections. First, an opening section, in which the interviewer introduces him/herself. Second, an introduction section that familiarizes the interviewee with the purpose of the interview and the research. The key questions are the main section of any interview. According to Gläser and Laudel (2009) there are different types of interview question that are depicted in Figure 4. Finally, a closing section which wraps up the interview and ask for further recommendations and so on (Myers and Newman 2007).

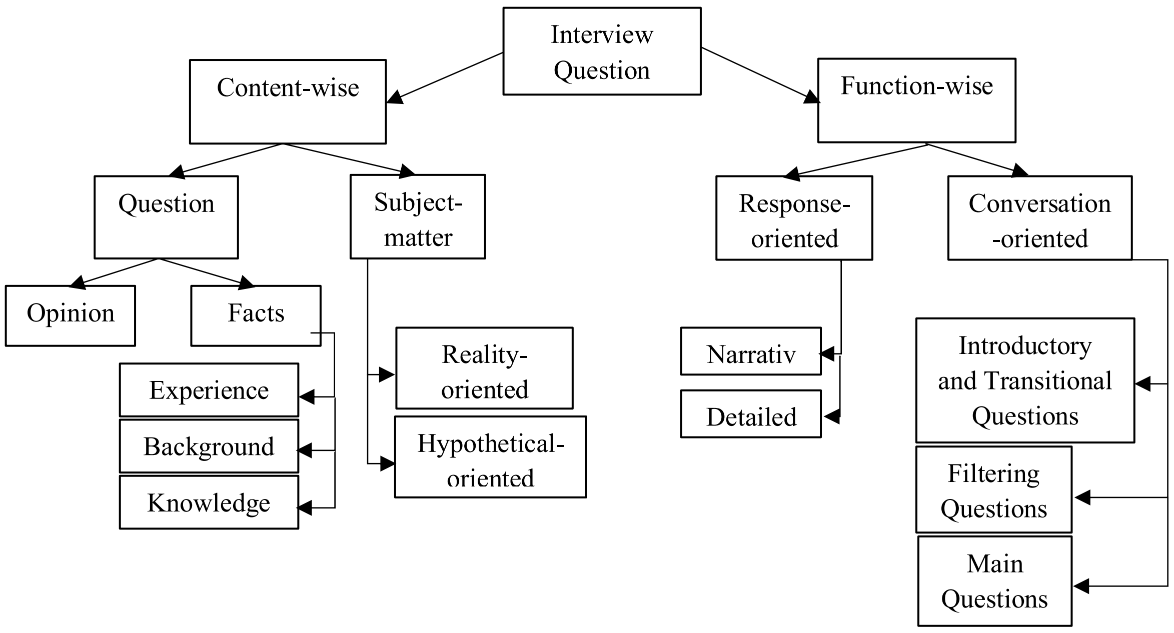


Figure 4 - Types of interview questions adapted from Gläser and Laudel (2009)

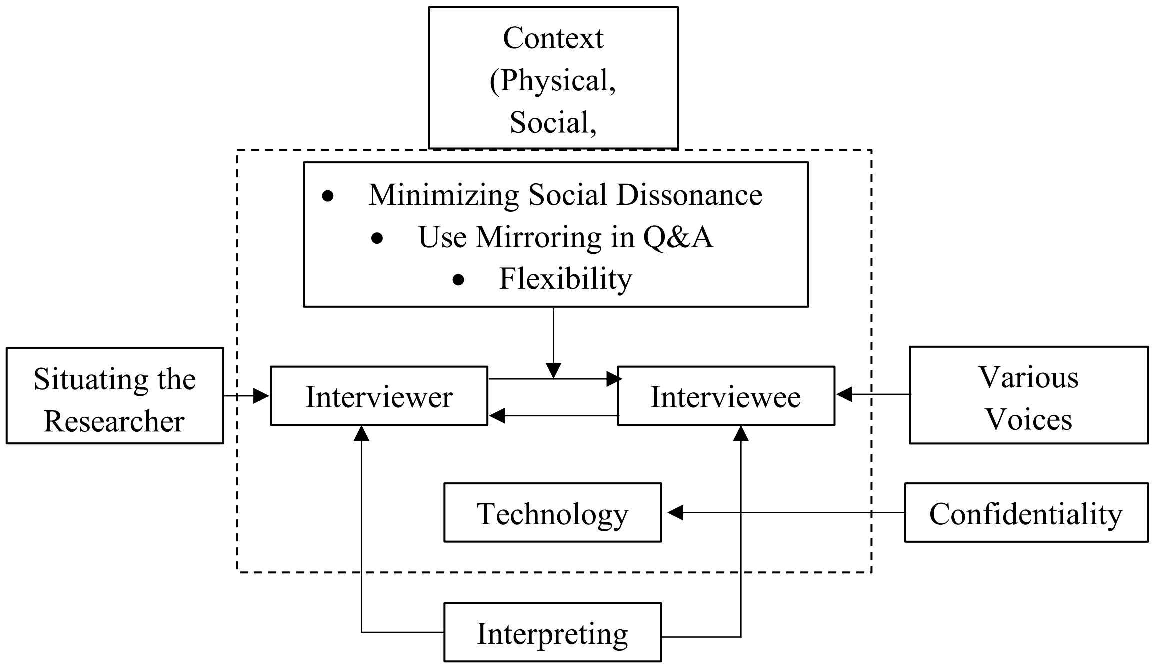


Figure 5 - Elements of qualitative expert interview according to Myers and Newman (2007)

Nonetheless, in semi-structured interviews, the script must not be over-prepared and leave room for flexibility and openness (Myers and Newman 2007). Moreover, we do not ask questions necessarily in the same order as they are planned, since the development of the interview highly depends on the interviewee and the provided inputs (Runeson and Höst 2009).

We also record an interview, based on the existing and technologies, if permitted by the interviewee. This is in addition to the notes that can be taken at the time of the interview. If it is recorded, the next step is to completely transcribe the conversation into the text that allows better and deeper analysis. The involved elements in an interview and major influential factors are illustrated in Figure 5.

Survey

Survey is a research method that uses standardized questionnaires to collect data. Surveys are able to expose unobservable data such as preferences, beliefs, and behaviors (Bhattacherjee 2012). Moreover, a survey can be conducted remotely that allows for a large target population to be studied. Small effects of variable can be better be identified in surveys with large sample set (Bhattacherjee 2012). In this dissertation, we perform an explanatory survey (Recker 2012) that aims to test hypotheses and relationship between variables (e.g. conflict and project success).

In a questionnaire survey, which is the case of this dissertation, a set of questions (items) are used to capture responses of subjects standardized and systematically (Bhattacherjee 2012). The questionnaire can be distributed in many ways such as mail, email, telephone, social media, etc. There are several common types of questions based on their responses (Bhattacherjee 2012). For example, a question may have a dichotomous response (e.g. yes or no), nominal response (i.e. selecting from an unordered list), ordinal response (i.e. selecting from an ordered list), interval-level response (e.g. Likert scales), and continuous response (e.g. asking for age of the participant).

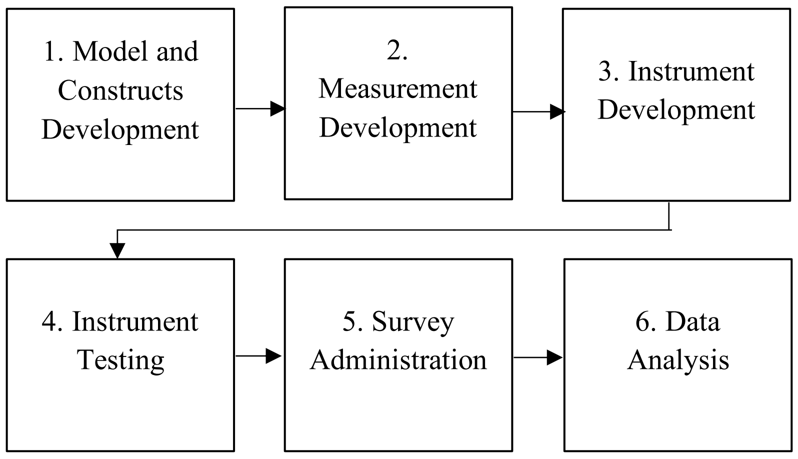

1. Model and Constructs Development
2. Measurement Development
3. Instrument Development
4. Instrument Testing
5. Survey Administration
6. Data Analysis

Figure 6 - Major steps for survey research according to Recker (2012)

Conducting a survey follows a several major steps that are depicted in Figure 6. We start by developing the model of the research based on prior theories and findings. Next, we develop a measurement for every construct. A construct is used to measure a notion that cannot be measured directly through observations (e.g. a particular type of conflict) and the more abstract the notion, the more challenging it is to measure it (Hinkin 1998). Subsequently, we build the complete instrument (e.g. questionnaire) including all measurements and other questions. In design of a survey various aspects must be considered such as wording of the questions, order of the questions, response time of the survey, and so on (Bhattacherjee 2012). To tackle such aspects and increase the validity of the survey, we need to perform a pre-test and/or a pilot test (Recker 2012). As the next step, we distribute the survey among the respondents and collect the data. Finally, we perform a proper analysis method on the collected data.

Nevertheless, several biases can harm the validity of a survey such as sampling bias, social desirability bias, and recall bias (Bhattacherjee 2012). Moreover, a survey is not able to achieve rich explanations and descriptions of a situation (Recker 2012). To cover such limitations, this dissertation conducts also in-depth expert interviews.

Structural Equation Modeling

Structural equation modelling (SEM) is a set of statistical procedures to test hypotheses through path analysis with latent variables (Bagozzi and Yi 2012; McDonald and Ho 2002). After receiving some hypotheses, models, and the collected data, SEM is able to provide three major outputs: numeric estimates about the models, logical implications about the models, and the degree to which data supports the implications (Kline 2015). SEM is considered as the state-of-the-art for high quality statistical survey analysis (Recker 2012).

In SEM, we distinguish between an observable variable and a latent variable (i.e. construct as mentioned in the survey design). An observed variable is collected directly from the subjects and is presented in the data. In contrast, a latent variable cannot be observed and measured directly. In case of this dissertation, different types of conflict and project success are latent variables, since we cannot directly measure to what extent a situation is conflicting or a project is successful, instead we search for presence or absence of some indicators (i.e. observed variables) such as meeting deadlines for the project success.

Several analyses are needed to make sure that the latent variables have been measured correctly. First, we analyze whether the indicators that were supposed to measure different latent variables are actually are corresponding to different latent variables. To this end, we perform principle factor analysis (Bryant and Yarnold 1995) through techniques such as investigating Eigenvalues and performing a Scree Test. Next, we analyze whether the used indicators are exactly measuring the latent variable that they were supposed to measure. For instance, in this dissertation, we analyzed whether the questions for measuring HRC are actually measuring HRC and not NHRC instead. Finally, after analyzing the model measurement, we can perform hypothesis assessment through various methods such as Root Mean Square Error of Approximation, Comparative Fit Index, and Tucker- Lewis Index.

Design Science

The design science approach has its roots in engineering and seeks for solving a problem through development of artifacts such as models, methods, and instantiations (i.e. prototypes) (Hevner et al. 2004).

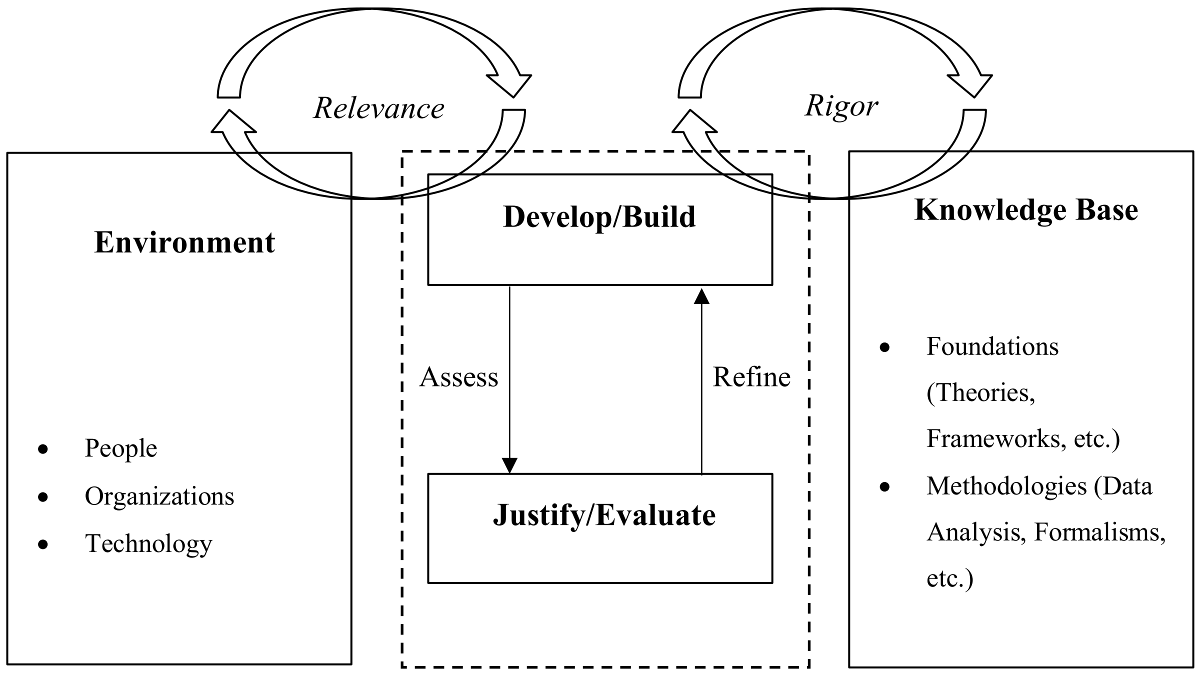


Figure 7 - Design science framework according to Hevner (2004)

Hevner et al. (2004) introduce the framework for design science (presented in Figure 7) and identify its main components. Accordingly, design science research finds the problem and needs from the environment. Similarly, development of a new artifact is based on prior knowledge such as theories, frameworks, and methodologies that are encapsulated in the knowledge base component. The environment defines the problem space (e.g. business needs) and creates the relevance cycle through which we receive the valuable problems and provide valuable solutions. Furthermore, the rigor cycle provides applicable knowledge for development of an artifact and receives new findings, methods, or even theories as additional knowledge to the knowledge base.

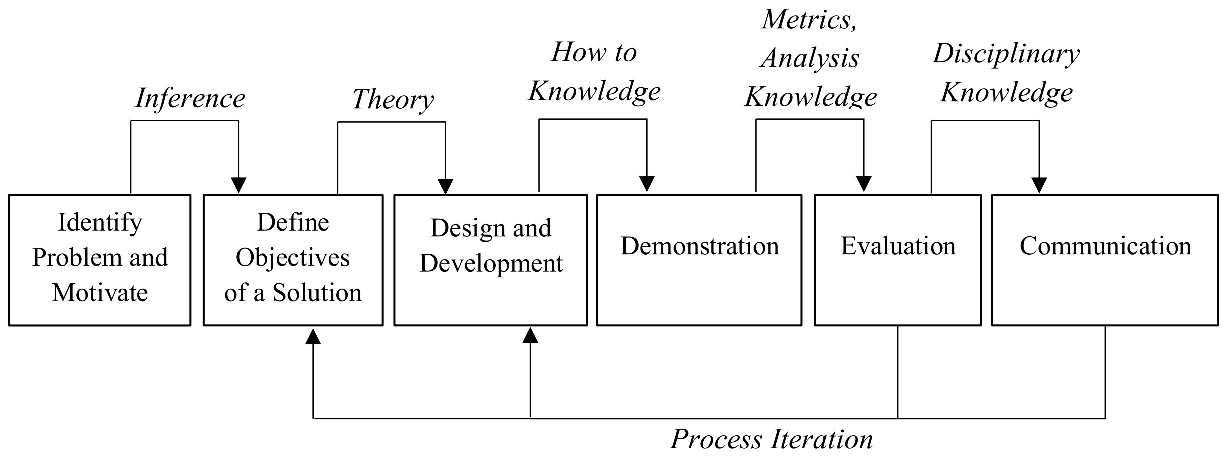


Figure 8 - Design science process according to Peffers et al. (2007)

The design science approach follows multiple iterations. In every iteration, the design of the artifact progress, some knowledge is created, communicated and some new knowledge whether from the relevance cycle or rigor cycle is transferred into the artifact design. The process of design science is presented in Figure 8.

According to Peffers et al. (2007), the process of design science starts with the identification of the problem, which comes from the environment component. The particular settings for the development of PSS, which is highly prone to conflicts, is the main motivation for the artifacts developed in this dissertation. By incorporating knowledge from both environment and knowledge base, we define the objectives of the solution. Subsequently, we design and develop the artifact. Next, we demonstrate, evaluate, and communicate it. During the evaluation and communication steps, we assess the artifact and its design. The feedback from these steps can be used iteratively to complete and refine the artifact.

The evaluation in the design science approach can be performed using various methods such as observational, analytical, experimental, test, and descriptive (Hevner et al. 2004). Venable (2006) distinguishes between artificial and naturalistic evaluations. The artificial evaluation describes a (nonrealistic) contrived situation, in which the solution of the design science should be applied. For instance, an exemplary case study is used to show how the solution can be applied in a particular situation. In contrast, a naturalistic evaluation investigates the extent, to which the solution can be effective in a real environment, e.g. through an experiment or case study. The studies of this dissertation mostly followed an artificial evaluation.

Part B1 (Main Publications)

P3: Understanding the Relationship of Conflict and Success in Software Development Projects

Title	Understanding the Relationship of Conflict and Success in Software Development Projects
Authors	Basirati, Mohammad R.* (mohammadreza.basirati@tum.de) Otasevic, Marko* (marko.otasevic@tum.de) Rajavi, Koushyar** (koushyar.rajavi@scheller.gatech.edu) Böhm, Markus* (markus.boehm@tum.de) Krcmar, Helmut* (krcmar@in.tum.de) * Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany ** Scheller College of Business - Georgia Institute of Technology, Georgia
Publication	Journal of Information and Software Technology (2020)
Status	Published
Contribution of the author	**Problem definition, research design, survey design, design and conducting interviews, data analysis, reporting**

Abstract –

Context: Software development incorporates numerous people with diverse expertise and expectations. This makes conflict a common phenomenon in software development. Besides human causes, many conflicts in soft- ware development root in the tools and processes. Moreover, the growing role of software in any type of system is increasing the heterogeneity in software projects. The number and variety of tools and processes are increasing. Nevertheless, the relationship between conflicts, particularly rooted in non-human elements, and software project success is still unclear.

Objective: We aim to understand the impact of conflict on the success of software development projects for different types of conflict and different environments. Particularly, we distinguish between human-rooted conflict (HRC) and non-human-rooted conflict (NHRC). Moreover, we investigate whether organization size and team size moderate the impact of conflict on software project success.

Methods: First, we conduct a survey and analyze it using structural equation modeling (SEM) to investigate any correlation between conflict and software project success. Second, we explore the reasons behind the relationship between conflict and software project success by conducting 13 semi-structured expert interviews.

Results: HRC is always a threat to software project success for any organization or team size. Based on the interviews, resolving an HRC is regularly problematic. On the other hand, NHRC is negatively correlated with software project success only in corporate organizations and small teams. High coordination overhead and dependency on tools and processes make NHRC more influential in corporate organizations. In contrast, overlooking non-human elements and lack of experienced individuals in smaller teams make them more vulnerable to NHRC.

Conclusion: While the detrimental impact of HRC is constant for software project success, NHRC can be controlled efficiently. Corporate organizations need to frequently improve the non-human elements in the development. Smaller teams should expect tools and processes to be significantly influential in their success.

Keywords: Conflict, Software project success, Software development, Non-human-rooted conflict, Organization size, Team size

P4: Towards Systematic Inconsistency Identification for Product-Service Systems

Title	Towards Systematic Inconsistency Identification for Product-Service Systems
Authors	Basirati, Mohammad R.* (mohammadreza.basirati@tum.de) Zou, Minjie* (minjie.zou@tum.de) Bauer, Harald* (harald.bauer@scheller.gatech.edu) Kattner, Nikolas* (nikolas.kattner@tum.de) Reinhart, Gunther* (reinhart@tum.de) Lindemann, Udo* (udo.lindemann@tum.de) Vogel-Heuser, Birgit* (vogel-heuser@tum.de) Böhm, Markus* (markus.boehm@tum.de) Krcmar, Helmut* (krcmar@in.tum.de) * Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Publication	International Design Conference (2018)
Status	Published
Contribution of the author	**Problem definition, research design, solution definition, reporting**

Abstract - Value shift towards services led to emergence of product-service systems (PSS) as intertwined products and services. PSS development requires collaborating teams with higher domain diversity to tackle service side as well as product side. Since every domain employs a particular set of tools and models, it is challenging to manage consistency among them. However, the PSS literature lacks approaches for managing inconsistency among various type of models. This study proposes a framework that supports establishing a systematic solution for inconsistency identification during PSS development.

Keywords: product-service systems (PSS), model-based engineering, modeling, systematic approach

Part B2 (Other Publications – Not Included in Review and Evaluation)

P1: IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation

Title	IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation
Authors	Basirati, Mohammad R.* (mohammadreza.basirati@tum.de) Weking, Jörg* (andreas.hein@tum.de) Hermes, Sebastian* (markus.boehm@tum.de) Böhm, Markus* (markus.boehm@tum.de) Krcmar, Helmut* (krcmar@in.tum.de) * Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Publication	Pacific Asia Conference of Information Systems (2019)
Status	Published
Contribution of the author	**Problem definition, research design, interview design, data analysis, reporting**

Abstract - Nowadays, product-service systems (PSS) as an integrated system of physical products and services play a crucial role in sustainable economies. In addition to high competitive global economy, emergence of new digital paradigms is supporting the shift towards servitization. Although the great potential of such paradigms are recognized by both practice and research, their implications for PSS is not clear yet. Particularly, features of Internet-of-Things (IoT) such as total connectedness and ubiquity of smart sensors and actuators provide various new opportunities for PSS. This study explores such opportunities by conducting structured literature review and 13 interviews. We formulate the findings into two folds. First, we introduce four degrees of IoT involvement in PSS business models and we elaborate the opportunities that they create for different types of PSS. Second, we present the key technologies and approaches, which IoT provides with regard to PSS lifecycle management.

Keywords: Product-Service System, Internet-of-Things, IoT Integration, Review

P2: Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation

Title	Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation
Authors	Basirati, Mohammad R.* (mohammadreza.basirati@tum.de) Weking, Jörg* (andreas.hein@tum.de) Hermes, Sebastian* (markus.boehm@tum.de) Böhm, Markus* (markus.boehm@tum.de) Krcmar, Helmut* (krcmar@in.tum.de) * Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Publication	Asia Pacific Journal of Information Systems (2019)
Status	Published
Contribution of the author	**Problem definition, research design, interview design, data analysis, reporting**

Abstract - Product–service systems (PSS), integrating physical products and services, currently play a crucial role in sustainable economies. In addition to the highly competitive global economy, the emergence of new digital paradigms is supporting the shift toward servitization. Although the great potential of such paradigms is recognized by both practice and research, their implications for PSS are not yet clear. In particular, features of Internet of Things (IoT), such as total connectedness and ubiquity of smart sensors and actuators, provide various new opportunities for PSS. This study explores such opportunities by conducting structured literature review and 13 interviews. We organize the findings in two folds: First, we introduce four degrees of IoT involvement in PSS business models and elaborate the opportunities that they create for different types of PSS. Second, we present the key technologies and approaches that IoT provides concerning PSS lifecycle management.

Keywords: Keywords: Product–Service System, Internet of Things, IoT Integration, Review, Expert Interview

P5: Introducing TRAILS: A Tool Supporting Traceability, Integration and Visualisation of Engineering Knowledge for Product-Service Systems Development

Title	Introducing TRAILS: A Tool Supporting Traceability, Integration and Visualisation of Engineering Knowledge for Product-Service Systems Development
Authors	Wolfenstetter, Thomas* (thomas.wolfenstetter@tum.de) Basirati, Mohammad R.* (mohammadreza.basirati@tum.de) Böhm, Markus* (markus.boehm@tum.de) Krcmar, Helmut* (krcmar@in.tum.de) * Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Publication	Journal of Systems and Software (2018)
Status	Published
Contribution of the author	**Development of artifact, reporting**

Abstract - Developing state of the art product service systems (PSS) requires the intense collaboration of different engineering domains, such as mechanical, software and service engineering. This can be a challenging task, since each engineering domain uses their own specification artefacts, software tools and data formats. However, to be able to seamlessly integrate the various components that constitute a PSS and also being able to provide comprehensive traceability throughout the entire solution life cycle it is essential to have a common representation of engineering data. To address this issue, we present TRAILS, a novel software tool that joins the heterogeneous artefacts, such as process models, requirements specifications or diagrams of the systems structure. For this purpose, our tool uses a semantic model integration ontology onto which various source formats can be mapped. Overall, our tool provides a wide range of features that supports engineers in ensuring traceability, avoiding system inconsistencies and putting collaborative engineering into practice. Subsequently, we show the practical implementation of our approach using the case study of a bike sharing system and discuss limitations as well as possibilities for future enhancement of TRAILS.

Keywords: Model-based systems engineering, Traceability, Product service systems, Model integration

P6: Facilitating Consistency of Business Model and Technical Models in Product-Service Systems Development: An Ontology Approach

Title	Facilitating Consistency of Business Model and Technical Models in Product-Service Systems Development: An Ontology Approach
Authors	Zou, Minjie* (minjie.zou@tum.de) Basirati, Mohammad R.* (mohammadreza.basirati@tum.de) Bauer, Harald* (harald.bauer@scheller.gatech.edu) Kattner, Nikolas* (nikolas.kattner@tum.de) Reinhart, Gunther* (reinhart@tum.de) Lindemann, Udo* (udo.lindemann@tum.de) Böhm, Markus* (markus.boehm@tum.de) Krcmar, Helmut* (krcmar@in.tum.de) Vogel-Heuser, Birgit* (vogel-heuser@tum.de) * Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Publication	9th IFAC Conference (2019)
Status	Published
Contribution of the author	**Problem definition, research design, solution definition, reporting**

Abstract - Due to the fast growing and ever-changing innovation cycles, industries are changing their strategies from a product-centric to service-centric approach, leading to the emergence of product-service systems (PSS) which integrate services into physical technical systems. In the model-based development of PSS, various models are employed by different stakeholders to represent their views on the system. However, there is a high diversity of these models in both forms and contents. Among others, business models and technical models come in different levels of abstraction, and thus, are hard to align with each other. In this study, we propose an approach to support PSS development by relating business models to technical models using ontology. Web ontology language (OWL) is employed to describe PSS knowledge, the Query Language SPARQL and Semantic Web Rule Language (SWRL) are used for consistency checking and reasoning potential inconsistencies.

Keywords: Product-service systems (PSS), model-based engineering, ontology, aligning business and technical models, model consistency.

PART C

4. Discussion

In this section, we discuss the findings of this dissertation concerning the related body of knowledge. We formulate the discussion according to the three research questions. First, we provide a summary of findings in a table for every research question. Subsequently, we explain how the findings of this dissertation extend the prior research.

PSS Development with IoT Integration

The following table presents knowledge nuggets of the first research question published in P1 and P2.

Table 9 - Summary of results for research question 1 (P1, P2)

Knowledge Nugget	Description
Framework of IoT-PSS business model	A holistic overview on possibilities of IoT integration into PSS business models based on three types of PSS (product-oriented, service-oriented, result oriented) and four levels of IoT integration (tracking, interacting, optimizing, transforming)
Framework of IoT-PSS lifecycle management	A holistic overview on core impacts of IoT on PSS development through notions such as closed-loop lifecycle management, collaboration, autonomy, digital twin, smart logistics, predictive maintenance, and remanufacturing
Concrete industrial cases for different IoT-PSS integrations	Briefly introducing twelve real-world industrial cases that have implemented different levels of IoT integration for PSS development
Challenges in IoT-PSS integration	The main challenges regarding to IoT-PSS integration such as collaboration of heterogenous knowledge experts, complex inter-organizational partnerships due to complexity of IoT implementation, alignment between simultaneous development of hardware and software

The significance of digital technologies is still rising, since they playing more roles in enabling service design, development, and delivery. The software has become ubiquitous and inseparable part of any system. This is highly reflected in IoT, as an interconnected sensing, data analysis, and actuating framework (Gubbi et al. 2013). IoT adds extra potential as well as challenges for PSS. Particularly, IoT enhances products and services with the connectivity of various heterogeneous components of a system such as PSS. Thus, as the first step towards understanding the conflicts in PSS development, we increase our knowledge of the environment from which conflicts emerge. The environment of PSS development is highly affected by software-intensive technologies, mainly, IoT.

IoT is mostly applied and integrated into existing systems. Moreover, PSS has to integrate IoT to leverage its benefits. Despite these facts, the IoT-PSS relationship is vague, particularly how IoT is integrated into PSS and influences PSS conceptualization and implementation. The related existing studies highly diverge in terms of concepts, terminology, and knowledge. Each study partly tackles a specific aspect of the PSS-IoT relationship. The connections between the findings of different studies are missing. The most related studies of IoT-PSS relationships, such as Seregni et al. (2016), Shih et al. (2016), and Zancul et al. (2016), are limited to low number of case analysis and hardly can be generalized. This dissertation fills this research gap by clarifying the integration of IoT in PSS with regard to PSS business models and PSS lifecycle management.

Particularly, such a contribution is of high importance for today, since we are just witnessing the starting phase of IoT applications; evolution and reaching to full potential of IoT is expected in the next upcoming years (Ardolino et al. 2016). Therefore, the related knowledge nuggets of this dissertation (presented in Table 9) pave the way for studying future challenges of IoT-PSS integration. Moreover, the practice still faces barriers to assess the benefits of IoT integration and compare them to the costs and challenges that IoT impose such as conflict. With this regard, we partly clarify the circumstances of integrating software-intensive technologies such as IoT. This supports a cost-benefit estimation that is more accurate and effective. In addition, the introduced frameworks of this section, enable PSS provides to better position themselves regarding IoT integration. They would be able to analyze the IoT potentials that they have not exploited and the implementation challenges that they will face.

Furthermore, we identified the implications of IoT for PSS lifecycle management. We provided a comprehensive overview of a wide range of related concepts and technologies. The proposed overview maps how every concept is related to other concepts and at which stage of the development. Most importantly, as the foundations for the next research question of this dissertation, we identified difficulties in IoT integration for PSS. The expert interviews confirmed that IoT adds extra complexity to any collaboration in PSS development. It challenges the existing methodologies as hardware and software components should be developed in intense coordination and harmony. Moreover, the vast amount of captured, generated, and collected data in IoT-driven PSS is highly challenging to be managed. Interoperability and compatibility among tools, artifacts, and data sources is a difficult goal to achieve.

Combining the first research question's knowledge nuggets (presented in Table 9) enlighten an overall situation, in which PSS is developed in the IoT era, particularly, how IoT can be exploited for PSS development and what new challenges it brings. If the IoT-PSS integration is realized correctly, it increases the reliability and smartness of PSS. Moreover, IoT can benefit PSS providers by shortening the development cycles and costs for example by utilizing IoT for a more effective maintenance.

The relationship of Conflict Types (HRC and NHRC) and Project Success

The following table summarizes the knowledge nuggets of the second research question, which are published in P3.

Table 10 – Summary of results for research question 2 (P3)

Knowledge Nugget	Description
New classification of human-rooted conflict (HRC) and non-human-rooted conflict (NHRC)	Distinguishing between conflicts rooted in human factors and conflicts rooted in non-human factors by highlighting the importance of non-human elements in software-intensive PSS development
Negative correlation of HRC and project success	HRC is detrimental for the project success, whether the organization is large or small, or the team is small or large.
Negative correlation of NHRC and project success with moderating effect of organization and team size	NHRC has negative impacts on the success of software projects, when the team is small or when the organization is large.
The reasons and mechanisms through which the conflict may influence project success negatively	Many reasons exposed by the interviews, the two most important ones are as following. First, coordination in large organizations significantly depends on non-human elements such as tools, processes, and structures. This leads to detrimental effects of NHRC in large organizations. Second, smaller software development teams suffer from the lack of experienced individuals and neglecting the importance of non-human factors. This causes NHRC to become highly problematic in smaller teams.

Conflict is a multidimensional concept. Consequently, the perspective from which we study conflict and how we classify conflict is of high significance. Every conflict classification allows us to analyze a particular set of factors, while we exclude other related dimensions. For instance, the distinction between relationship conflict and task conflict focuses on the nature of decisions, in which conflicts emerge.

In this dissertation, we acknowledge the rising importance of non-human elements, particularly for conflicts. Moreover, the prior classifications of conflict overlooked the role of non-human elements. While our findings from a prior study, presented in Section Conflicting Elements0, showed the elements constituting conflicts may be of human and non-human nature. Thus, we introduced a new classification of conflicts that distinguishes between HRC and NHRC. Such an emphasized distinction provides a new perspective that highlights the equivalent role of human and non-human elements in causing conflicts. Consequently, the new classification has direct implications for many aspects of conflict management that we discuss in the following. The new classification and its implications are briefly presented in Table 10.

On one hand, human-rooted elements are commonly challenging to change and improve. For instance, background, culture, language, and personality of individuals cannot be altered. Individuals are mostly reluctant or in some cases, intolerant about changing their opinions and beliefs. On the other hand, non-human elements inherently can be changed in a way easier. For example, tools can be exchanged, processes and methodologies can be adjusted, and the formats of documents can be redefined. Thus, we should take different strategies for the resolution of HRC and NHRC. For example, we might prefer to avoid HRC, while we tolerate some degree of NHRC. Moreover, in a situation of limited resources (e.g. lack of budget or time), we can better decide how to allocate resources to a conflicting situation. Besides, recognizing HRC and NHRC prevent us to mistakenly waste solutions that do not fit the type of conflict. For example, we can avoid getting into a fight with an individual or a team because of non-aligned tools. Similarly, we might not mistakenly reshape teams or change processes simply because two individuals' personalities and beliefs do not allow them to collaborate.

Furthermore, we investigated whether HRC and NHRC affect project success differently. Our analysis confirmed such a difference. While we found HRC always negatively correlated with project success, organization size and team size could moderate the detrimental effects of NHRC. Further in-depth expert interviews also revealed how HRC and NHRC affect a project differently. We could identify two major reasons. First, as explained above, since human elements are fixed and hard-to-adjust, other factors (e.g. team and organizational) cannot easily change them to mitigate their negative effects. In contrast, non-human elements change according to other factors such as the team or organization. Second, while any project, team, or organization is always dependent on its contributing individuals, the dependence on non-human elements can vary hugely in different projects or teams. For example, we found in large organizations the information flow is highly dependent on tools and formal processes. Consequently, NHRC is negatively correlated with project success in such settings. Acknowledging the difference implications of HRC and NHRC can support us to make the most effective decisions in times of a conflicting situation. We would be able to better analyze the situation based on the type of conflict, organization, and team.

We also found that NHRC is only influencing project success negatively in small teams (less than ten people). This was a surprise for our interview subjects as they did expect the opposite results similar to the effect of organization size. This shows that there are still many unknowns about the dynamics of conflicts, how they emerged and get resolved.

Such implications of HRC and NHRC distinction can hugely improve and facilitate PSS development. Because PSS development is mostly practiced through cross-functional teams spread over different departments and organizations. In such complex settings, conflicts are also complex and prevalent. A correct understanding of different conflict types' nature enables us to manage the complexity of conflicts, shorten the development time, and increase the quality of the work and the final system.

Mechanisms to Identify NHRC in PSS Development

The following table summarizes the results of research question three, which are published in P4, P5, and P6.

Table 11 - Summary of results for research question 3 (P4-P6)

Knowledge Nugget	Description
Technical conflict/inconsistency types	A detailed classification of low-level inconsistencies in PSS development based on the relation type of two information source (e.g. refinement or satisfaction) and abstraction level of inconsistency (e.g. conventional, project-specific, or domain-specific)
Framework of inconsistency identification in PSS development	A thorough process of inconsistency identification in PSS development depicted on three meta-model, model, and reality abstraction levels
Influencing parameters of inconsistency identification in PSS development	Two group of parameters: problem-side parameters such as type of inconsistency and solution-side parameters such as degree of automation for inconsistency identification
A tool for model-based traceability and integration of engineering knowledge in PSS	A tool that can assist integration of various engineering knowledge to trace them and identify the inconsistencies among them
An approach for alignment between business and service-side with technical and product-side	An ontology approach based on SPARQL and semantic web technologies to develop an ontology, which can be used to infer about the relation between service-side and product-side of a PSS that enables identification of conflicts

After increasing our knowledge about the circumstances in PSS development, different types of conflicts, and impacts of different conflict types, we focus on identifying conflicts, particularly low-level NHRC, i.e. technical inconsistencies, in PSS development. To this end, we developed a framework, a tool, and an approach (presented in Table 11).

PSS development highly depends on the type of products and services, which it integrates. For instance, a car-sharing PSS follows a very different process than a pay-per-usage printing system. Most importantly, the number and type of involved elements vary. In a car-sharing PSS, a significant number of different elements from various organizations and departments are interconnected. While, a printing PSS provider has a much lower number of elements, teams, and departments involved. Moreover, the design of products or software parts highly differ. A use-oriented PSS may allocate its main resources to software and service development, while a product-oriented PSS mostly focuses on its products enhanced with some software and service. Thus, we cannot provide a general solution for conflict identification in PSS. Instead, we provide a holistic framework that constitutes the general process, which any PSS needs to follow to identify inconsistencies. Moreover, we strived to design a systematic approach for inconsistency identification, since the industrial settings of PSS, as explored in the first research question, consists of numerous interconnected elements. Without a systematic approach consisting of formal processes and parameters, it is impossible to identify inconsistencies with high recall. In addition to defining a general process for systematic identification of inconsistencies in PSS, we also shed light on different types of inconsistencies and influential parameters in realizing a customized conflict identification for every PSS.

5. Implications

We discuss the high-level implications of this dissertation in two folds. First, we explain the implications for theory. Second, we elaborate on how this dissertation contributes to the practice. As we used a mixed-method approach, we achieved results of various forms from exploratory to confirmatory with the focus of finding the best possible solutions. Thus, we believe this dissertation provides rich and influential implications for both theory and practice.

Implications for Theory

The results of this dissertation contribute mainly to two major research streams: PSS literature and the conflict literature. In various ways, we contribute to the PSS literature such as providing new insights based on empirical studies and proposing new conflict identification mechanisms. We also contribute significantly to the studies on conflict management by introducing a new classification and analyzing the impacts of the new conflict types on project success. In the following, we dive deeper into every contribution and explain them in detail.

First, the PSS literature lacks empirical studies (Annarelli et al. 2016). This dissertation tackles this gap by conducting in-depth qualitative expert interviews that provide new information on how PSS can be conceptualized and developed with advanced digital technologies such as IoT. Moreover, the existing empirical studies in PSS literature are mostly single case studies (Shih et al. 2016; Zancul et al. 2016). Therefore, we also contribute to research by providing experiences of experts from a wide range of PSS cases.

Second, we contribute to the emerging stream of IoT-PSS integration/alignment studies. The studies in this new stream of research are highly diverse and there is no consensus among the studies concerning the big picture, related concepts and dimensions, and terminology. This dissertation covers this gap by proposing two holistic frameworks that extend and consolidate the existing knowledge on the relationship between PSS and IoT. Consequently, the frameworks establish the foundation for further research.

Third, we contribute to the PSS literature by proposing new mechanisms for identification of inconsistency in PSS development. The existing related studies have only addressed the limited scope of conflicts. Shimomura and Hara (2010) mostly focused on identifying differences in names and conventions in PSS development. The study of Feldmann et al. (2015) is merely about manufacturing products and overlook the service and business side of PSS. In contrast, Song and Sakao (2016) only concentrate on the service side of PSS with a limited scope. The combination of the proposed mechanisms of this dissertation makes an almost complete picture of how inconsistencies in PSS should be identified. The mechanisms consist of a systematic approach formulated in formal holistic processes and parameters (p4) that can be customized and implemented for any PSS. The inconsistencies among technical artifacts can be identified using the proposed tool (P5) and the alignment between the technical product-side and service-side can be tackled using the proposed ontology-based approach (P6). Close alignment between different PSS components is necessary for effective and optimized design and development of PSS (Zacharewicz et al. 2017).

Fourth, we contribute to the conflict literature by extending the classifications of conflict. The prior research on conflict hardly distinguished between conflicts rooted in human causes and non-human causes. Nevertheless, such distinction is highly relevant for PSS development as the number of connected elements, whether human or non-human, increases significantly. The digital technologies used and developed in PSS impose close collaboration of various domains and teams through heterogeneous processes, tools, and artifacts (Song 2017; Vasantha et al. 2012; Wiesner et al. 2017). Consequently, the role of non-human elements in PSS development becomes substantial. We tackle this gap by introducing HRC and NHRC types. Acknowledging the distinction between HRC and NHRC and their dissimilar natures helps us to better understand the dynamic of conflicts.

Fifth, as another contribution to the studies on conflict, we provide new empirical evidence on the negative effects of HRC. Our findings confirm previous studies regarding the detrimental effects of HRC (Guang-dong 2013; He 2007), which showed organization and team size do not moderate negative HRC. However, our results question the positive effects of conflicts such as better ideation and learning (Chen et al. 2004; Liang et al. 2009). Because we found both HRC and NHRC as negatively correlated with project success. According to the in-depth interviews, we found that probably, the negative effects reported in the survey are short term, while in the long term, conflicts can be leveraged to highlight weaknesses and lessons-learned.

Sixth, conflict studies have mostly focused on group-level implications of conflict such as team performance. In this dissertation, we shift to another dimension with a higher granularity, which is the project success. Therefore, our findings of the relationship between HRC and NHRC and project success contributes to not only the conflict studies but also the research on project success factors.

Seventh, we deepen our knowledge about different aspects of conflict by exposing the moderating effects of two contextual factors on conflicts' impact, namely, team size and organization size. We contribute to the conflict literature by providing new empirical evidence on how conflict's consequences may change with regard to team size and organization size.

In summary, the results of this dissertation have several implications for the PSS research as well as the conflict research. We extend the PSS studies by not only providing new empirical insights but also proposing a set of mechanisms to identify inconsistencies in PSS development. We also introduce new types of conflicts and provide a deep analysis of how they influence project success.

Implications for Practice

This dissertation put a high emphasis on practical contribution. This is reflected in the research methodology and the results. The mixed-method approach allowed us to pay more attention to problem-solving than merely providing rigorous research findings that may not be relevant to practice. Moreover, we strived to tackle different aspects of conflict in PSS development and the results vary from empirical findings to, frameworks, methods, and tools. In the following, we present the implications of this dissertation for the practice.

Based on our findings, the practice has not fully exploited the capabilities of IoT for advancements in PSS design and development. For practitioners, it is still unclear how to position and integrate IoT into PSS. The proposed IoT-PSS frameworks of this dissertation can support managers in assessing the potentials of IoT for advancing their business models as well as enhancing their developments. For instance, a PSS provider can evaluate to what extent IoT can create value for the PSS and what technologies and practices are needed for development of such PSS.

Furthermore, as a major contribution, the practice can employ the set of mechanisms provided by this dissertation for inconsistency identification in PSS development. The introduced systematic approach for inconsistency identification can be fully customized to any PSS development. We introduced different types of conflicts and inconsistencies that should be tackled and how an identification process can be adjusted to identify them according to several main influential parameters. As the PSS development is highly complex, such a systematic approach is of high importance and if implemented correctly, it can save considerable amount of cost and time for PSS providers. Particularly, inconsistencies if cannot be detected early, they can lead to great failures (Gervasi and Zowghi 2005).

Our findings about the HRC, NHRC, and their correlation with project success implicate several managerial concerns. First, practitioners need to recognize the importance of conflicts caused by human factors. Since, the presence of HRC has negative consequences in any situation and almost no benefits. Thus, managers must strive for a constructive environment to mitigate the negative effects of HRC. We found that organizational culture plays a big role in how conflict resolution. Second, managers in corporate organizations need to recognize the critical role of non-human factors such as tools and processes. They should expect more frequent NHRC. Hence, they need to be prepared by aligning the tools, processes, and planning solutions for emergent conflicts. To this end, they can use the mechanisms introduced in this dissertation such as the ontology-based solution, the model-integration tool, and the systematic inconsistency identification approach. Besides, for corporates, we recommend continuous improvement of non-human elements such as processes, tools and mechanisms for increasing awareness among employees. On the other hand, smaller organizations such as start-ups and medium-sized companies do not need to spend extensive amount of time and money for defining workflow processes and advanced tools. However, the practitioners in small teams should not underestimate the extent to which, non-human elements can create difficulties and harm their success. Since, our findings showed that NHRC in small teams are negatively correlated with project success.

To summarize, this dissertation provides a variety of findings that have significant managerial implications. The created knowledge is formulated into frameworks, classification, and methods. Managers can exploit the created knowledge and apply the frameworks and methods to improve PSS development in a way that the development is more efficient, in terms of time and budget. Moreover, more effective conflict management, enabled by findings of this dissertation, can benefit individuals and the overall organization atmosphere, since conflicts can directly harm the individuals’ feelings and disturb the work environment. Furthermore, we directly contribute to practice by introducing a set of practical mechanisms for inconsistency identification, which can be easily customized and implemented for every PSS development.

6. Limitations

No research is complete without determining its limitations. The results of this dissertation are subject to several limitations. The limitations emerge from the general research approach of this dissertation, the employed research methods, data sources, and how we interpreted them. In the following, we explain several limitations of this study in detail.

First, findings of a structured literature review are subject to three main limitations: the search process, the subjective mind of the researcher, who selects the relevancy of the paper and extract the findings, and the publication bias. The reason that we performed a structured literature review was to mitigate such limitations through a systematic process (Kitchenham et al. 2009). Moreover, the applied literature review conducted in P1 is performed by two researchers to enhance the validity of the data analysis.

Second, we conducted a high number of qualitative expert interviews. The data collected through interviews is biased through the limited experiences of the interviewees and their beliefs. Therefore, they lack generalizability to other individuals and settings (Johnson and Onwuegbuzie 2004). To mitigate this issue, we strived to perform the interview with a wide range of unrelated individuals to collect multiple (contradicting) perspectives (Schultze and Avital 2011). Moreover, an interview may be affected by various factors such as lack of trust, lack of time, the ambiguity of language, and so on (Myers and Newman 2007). In addition to validity threats during data collection, interviews are also subject to the general limitations of qualitative data analysis such as coding bias.

Third, this dissertation employed a survey study analyzed through SEM techniques. A survey is highly influenced by its questionnaire and survey participants. Although we quantitatively evaluated the way we measured our constructs, we cannot in any way confirm or prove that we actually measured what we aimed for. Moreover, the wording and sequencing of the questions might have had impacts on how the participants perceived the questions, which may have influenced their responses. More importantly, our survey had a relatively low number of participants (about 110), which limits the generalizability of its results. As a minor issue, the way we distributed the survey might have also affected the population under investigation. Since we only distributed the survey through the internet, it could exclude the experienced but old-fashioned experts who are not active on social media websites. Nevertheless, we strived to overcome these shortcomings through a new round of qualitative in-depth expert interviews. The combined results are presented in P3.

To develop new methods and tools we employed a design science approach. In design science, we need a trade-off between the rigor and relevance of the research and in many cases, depending on the goal of the research we lean more in one direction (Hevner et al. 2004). In this dissertation, we focused more on the relevancy of the results for the practice. Hence, to some extent, we are subject to limitations from the rigor side. As the main limitation, we evaluated our methods artificially (Venable 2006), i.e. we did not have the chance to empirically assess the effectiveness of the proposed methods and tools in real-world case studies. This harms the generalizability of our methods. Nonetheless, the iterative nature of design science appreciates any advancements in the design and development of the artifacts. There is no perfect design and developed artifact in design science as we are always improving our results iteratively according to new settings and contexts.

Overall, this dissertation is limited to its definitions of the main concepts. The results highly depend on how we defined the concept of conflict and the conflict types of NHRC and HRC. The focus of this dissertation was to achieve high relevance for practice. For instance, the classification of conflicts into NHRC and HRC can be comprehended and employed by practitioners much easier than the complicated types of task and relationship conflict. Nevertheless, our definitions of conflict and its types determine the scope of our results and their generalizability. Particularly, as the phenomenon of conflict is multidimensional, any definition or classification tackles several dimensions and overlooks some other aspects. Therefore, we do no claim that this dissertation could address all aspects of conflict and is limited to its scope determined by its definitions.

7. Future Research

Our findings increased our understanding of conflict, particularly in PSS development. However, the created knowledge triggered new open questions that future research needs to answer.

Regarding the implications of new digital technologies for PSS, there is still huge room for advancements. We lack mechanisms to analyze and estimate IoT adoption in terms of quantitative monetary parameters. Based on our findings, the practitioners cannot evaluate the benefits and costs of IoT in a realistic way. Thus, future research can develop innovative mechanisms for more precise assessment of IoT's implications for PSS.

As conflict is a multidimensional phenomenon that affects and is affected by various elements, many questions are still open. First, future research should answer the following question: what main factors determine the impact of conflicts? This dissertation only investigated the moderating effect of organization size and team size, which both are static variables, i.e. we cannot change them. Having more knowledge on the factors, which can mitigate or control the negative effects of conflict helps significantly to improve the work situation, remove the development barriers, and save time and money.

Furthermore, resolution strategies have a big influence on the consequences of a conflict. Nevertheless, in this dissertation, we only focused on identification phase of conflict management and resolution aspects are out of scope of this work. Therefore, we encourage future studies to investigate the extent, to which various mechanisms and strategies resolve both HRC and NHRC.

Similarly, future research can dive deep into conflict resolution in PSS development. Particularly, after the identification of conflicts, what are the proper mechanisms to resolve conflicts between the technical-side and service-side. Moreover, we need to what actions and considerations are necessary to keep the PSS development in the state of consistency. To answer this question, we also need to know to what extent, consistency is desirable and necessary for PSS development. Finally, although we emphasized on importance of NHRC in PSS development, we lack empirical findings on how often different types of conflict occur in practice. Future research can search for frequent conflicting situations in PSS development. This is of high importance for further development of tools and mechanisms for conflict resolution.

8. Conclusion

The growing globalized competitiveness as well as ecological considerations led to the emergence of product-service systems (PSS) as an integrated bundle of products and services. The growth of PSS is accelerated with advancements in complex digital technologies such as IoT. Software-intensive systems allowed the integration of various components from service-side and product-side enhanced by IoT elements. Nevertheless, such integration increased the heterogeneity and complexity of PSS enormously. This led to frequent emergence of conflicts in PSS development, not only among individuals from different domains and teams, but also between any two elements such as tools, artifacts, or even processes.

This dissertation aimed to tackle this problem by increasing our knowledge about conflicts in PSS development. To this end, we employed a mixed-method approach. We used structured literature review, expert interviews, survey study (analyzed using SEM), and design science approach. Reflected on three conducting research questions, we provided insight on how IoT affects PSS development as circumstances, in which conflicts emerge, the relationship between different types of conflicts and project success, and a set of mechanisms to identify conflicts in PSS development systematically. Moreover, we introduced a new classification of conflicts into human-rooted conflict (HRC) and non-human-rooted conflict (NHRC). Using such a classification, we emphasized the importance of non-human elements in PSS development such as tools, documents, models, processes, and so on. The findings also showed that NHRC, similar to HRC, is negatively correlated with project success. However, in contrast to HRC, its negative effects can be moderated in different settings.

We contributed to theory by providing new empirical insights on PSS development, conflict types, the relation between conflict and project success, and the dynamics of conflicts and how they influence the project. More importantly, we introduced a new classification of conflicts that can be further be used to analyze the complex nature of conflicts. Our developed mechanisms for conflict identification in PSS also provide the foundations for more advanced conflict management tools and methods. Furthermore, we have significant implications for practice, particularly, as we followed a pragmatism strategy to increase the relevancy of our results for the practice. The new conflict classification and insights on the conflict have huge managerial implications, as managers can employ such knowledge to improve conflict management and reduce wasted time and money. Moreover, the proposed set of mechanisms for systematic identification of conflicts in PSS development can be directly be realized and customized for PSS providers. More effective identification of conflicts in PSS development can considerably avoid extra costs and delays and in some cases prevent large failures.

Future research can dive deeper into various aspects of conflict, partially addressed in this dissertation. Particularly, how different factors can mitigate or control the impact of conflicts, the effectiveness of different conflict resolution strategies in PSS development, and mechanisms for conflict resolution for PSS.

References

Aldekhail, M., Chikh, A., and Ziani, D. 2016. "Software Requirements Conflict Identification: Review and Recommendations," *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS* (7:10), pp. 326-335.

Alexopoulos, K., Koukas, S., Boli, N., and Mourtzis, D. 2018. "Architecture and Development of an Industrial Internet of Things Framework for Realizing Services in Industrial Product Service Systems," *Procedia CIRP* (72), pp. 880-885.

Annarelli, A., Battistella, C., and Nonino, F. 2016. "Product Service System: A Conceptual Framework from a Systematic Review," *Journal of Cleaner Production* (139), pp. 1011-1032.

Aranda, G. N., Vizcaíno, A., and Piattini, M. 2010. "A Framework to Improve Communication During the Requirements Elicitation Process in Gsd Projects," *Requirements Engineering* (15:4), pp. 397-417.

Ardolino, M., Saccani, N., Gaiardelli, P., and Rapaccini, M. 2016. "Exploring the Key Enabling Role of Digital Technologies for Pss Offerings," *8th CIRP IPSS CONFERENCE-Product-Service Systems across Life Cycle, 2016*: Elsevier, pp. 561-566.

Azadegan, A., Papamichail, K. N., and Sampaio, P. 2013. "Applying Collaborative Process Design to User Requirements Elicitation: A Case Study," *Computers in Industry* (64:7), pp. 798-812.

Baccarini, D. 1996. "The Concept of Project Complexity—a Review," *International journal of project management* (14:4), pp. 201-204.

Bagozzi, R. P., and Yi, Y. 2012. "Specification, Evaluation, and Interpretation of Structural Equation Models," *Journal of the academy of marketing science* (40:1), pp. 8-34.

Baines, T. S., Lightfoot, H. W., Evans, S., Neely, A., Greenough, R., Peppard, J., Roy, R., Shehab, E., Braganza, A., and Tiwari, A. 2007. "State-of-the-Art in Product-Service Systems," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* (221:10), pp. 1543-1552.

Basirati, M. R., Hermes, S., Weking, J., Böhm, M., and Krcmar, H. 2019a. *Iot as Pss Enabler: Exploring Opportunities for Conceptualization and Implementation*.

Basirati, M. R., Otasevic, M., Rajavi, K., Böhm, M., and Krcmar, H. 2020. "Understanding the Relationship of Conflict and Success in Software Development Projects," *Information and Software Technology* (126), p. 106331.

Basirati, M. R., Weking, J., Hermes, S., Böhm, M., and Krcmar, H. 2019b. "Exploring Opportunities of Iot for Product–Service System Conceptualization and Implementation," *Asia Pacific Journal of Information Systems* (29), pp. 524-546.

Basirati, M. R., Zou, M., Bauer, H., Kattner, N., Reinhart, G., Lindemann, U., Böhm, M., Krcmar, H., and Vogel-Heuser, B. 2018. *Towards Systematic Inconsistency Identification for Product Service Systems*.

Behfar, K. J., Mannix, E. A., Peterson, R. S., and Trochim, W. M. 2011. "Conflict in Small Groups: The Meaning and Consequences of Process Conflict," *Small Group Research* (42:2), pp. 127-176.

Berenbach, B. 2006. "Impact of Organizational Structure on Distributed Requirements Engineering Processes: Lessons Learned," in: *Proceedings of the 2006 international workshop on Global software development for the practitioner*. Shanghai, China: ACM, pp. 15-19.

Berkovich, M., Leimeister, J. M., and Krcmar, H. 2011. "Requirements Engineering for Product Service Systems," *Business & Information Systems Engineering* (3:6), pp. 369-380.

Beuren, F. H., Ferreira, M. G. G., and Miguel, P. A. C. 2013. "Product-Service Systems: A Literature Review on Integrated Products and Services," *Journal of cleaner production* (47), pp. 222-231.

Bhat, J. M., Gupta, M., and Murthy, S. N. 2006. "Overcoming Requirements Engineering Challenges: Lessons from Offshore Outsourcing," *IEEE Software* (23:5), pp. 38-44.

Bhattacherjee, A. 2012. "Social Science Research: Principles, Methods, and Practices,").

Bjarnason, E., and Sharp, H. 2015. "The Role of Distances in Requirements Communication: A Case Study," *Requirements Engineering*), pp. 1-26.

Bostrom, R. P., and Heinen, J. S. 1977. "Mis Problems and Failures: A Socio-Technical Perspective, Part Ii: The Application of Socio-Technical Theory," *MIS Quarterly* (1:4), pp. 11-28.

Bressanelli, G., Adrodegari, F., Perona, M., and Saccani, N. 2018. "The Role of Digital Technologies to Overcome Circular Economy Challenges in Pss Business Models: An Exploratory Case Study," *Procedia CIRP* (73:2018), pp. 216-221.

Brocke, J. v., Simons, A., Niehaves, B., Niehaves, B., Reimer, K., Plattfaut, R., and Cleven, A. 2009. "Reconstructing the Giant: On the Importance of Rigour in Documenting the Literature Search Process,").

Bryant, F. B., and Yarnold, P. R. 1995. "Principal-Components Analysis and Exploratory and Confirmatory Factor Analysis,").

Chakraborty, S., Rosenkranz, C., and Dehlinger, J. 2015. "Getting to the Shalls: Facilitating Sensemaking in Requirements Engineering," *ACM Trans. Manage. Inf. Syst.* (5:3), pp. 1-30.

Chen, H.-G., Jiang, J. J., Chen, J.-C., and Shim, J. 2004. "The Impacts of Conflicts on Requirements Uncertainty and Project Performance," *Journal of International Technology and Information Management* (13:3), p. 2.

Coughlan, J., and Macredie, D. R. 2002. "Effective Communication in Requirements Elicitation: A Comparison of Methodologies," *Requirements Engineering* (7:2), pp. 47-60.

Damian, D., and Zowghi, D. 2003. "Re Challenges in Multi-Site Software Development Organisations," *Requirements Engineering* (8:3), pp. 149-160.

Daneva, M., Marczak, S., and Herrmann, A. 2014. "Engineering of Quality Requirements as Perceived by near-Shore Development Centers' Architects in Eastern Europe: The Hole in the Whole," in: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. Torino, Italy: ACM, pp. 1-10.

Daspit, J., Justice Tillman, C., Boyd, N. G., and Mckee, V. 2013. "Cross-Functional Team Effectiveness: An Examination of Internal Team Environment, Shared Leadership, and Cohesion Influences," *Team Performance Management: An International Journal* (19:1/2), pp. 34-56.

De Dreu, C. K., and Weingart, L. R. 2003. "Task Versus Relationship Conflict, Team Performance, and Team Member Satisfaction: A Meta-Analysis," *Journal of applied Psychology* (88:4), p. 741.

Easterbrook, S., and Nuseibeh, B. 1996. "Using Viewpoints for Inconsistency Management," *Software Engineering Journal* (11:1), pp. 31-43.

Easterbrook, S. M., Beck, E. E., Goodlet, J. S., Plowman, L., Sharples, M., and Wood, C. C. 1993. "A Survey of Empirical Studies of Conflict," in *Cscw: Cooperation or Conflict?* Springer, pp. 1-68.

Evans, S., Partidário, P. J., and Lambert, J. 2007. "Industrialization as a Key Element of Sustainable Product-Service Solutions," *International Journal of Production Research* (45:18-19), pp. 4225-4246.

Feldmann, S., Herzig, S. J., Kernschmidt, K., Wolfenstetter, T., Kammerl, D., Qamar, A., Lindemann, U., Krcmar, H., Paredis, C. J., and Vogel-Heuser, B. 2015. "Towards Effective Management of Inconsistencies in Model-Based Engineering of Automated Production Systems," *IFAC-PapersOnLine* (48:3), pp. 916-923.

Geraldi, J., Maylor, H., and Williams, T. 2011. "Now, Let's Make It Really Complex (Complicated) a Systematic Review of the Complexities of Projects," *International Journal of Operations & Production Management* (31:9), pp. 966-990.

Gervasi, V., and Zowghi, D. 2005. "Reasoning About Inconsistencies in Natural Language Requirements," *ACM Transactions on Software Engineering and Methodology (TOSEM)* (14:3), pp. 277-330.

Ghorpade, J., Lackritz, J., and Singh, G. 2011. "Personality as a Moderator of the Relationship between Role Conflict, Role Ambiguity, and Burnout," *Journal of Applied Social Psychology* (41:6), pp. 1275-1298.

Gläser, J., and Laudel, G. 2009. *Experteninterviews Und Qualitative Inhaltsanalyse: Als Instrumente Rekonstruierender Untersuchungen*. Springer-Verlag.

Goedkoop, M. J., Van Halen, C. J., Te Riele, H. R., and Rommens, P. J. 1999. "Product Service Systems, Ecological and Economic Basics," *Report for Dutch Ministries of environment (VROM) and economic affairs (EZ)* (36:1), pp. 1-122.

Greer, L. L., and Jehn, K. A. 2007. "Chapter 2 the Pivotal Role of Negative Affect in Understanding the Effects of Process Conflict on Group Performance," in *Affect and Groups*. Emerald Group Publishing Limited, pp. 21-43.

Greer, L. L., Jehn, K. A., and Mannix, E. A. 2008. "Conflict Transformation: A Longitudinal Investigation of the Relationships between Different Types of Intragroup Conflict and the Moderating Role of Conflict Resolution," *Small group research* (39:3), pp. 278-302.

Guang-dong, W. 2013. "The Relationship between Project Team Dynamic Feature, Conflict Dimension and Project Success--an Empirical Research from Shanghai, China," *Pakistan Journal of Statistics* (29:6).

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. 2013. "Internet of Things (Iot): A Vision, Architectural Elements, and Future Directions," *Future generation computer systems* (29:7), pp. 1645-1660.

Halme, M., Anttonen, M., Hrauda, G., and Kortman, J. 2006. "Sustainability Evaluation of European Household Services," *Journal of Cleaner Production* (14:17), pp. 1529-1540.

Hanisch, J., and Corbitt, B. 2007. "Impediments to Requirements Engineering During Global Software Development," *European Journal of Information Systems* (16:6), pp. 793-805.

He, J. 2007. "The Moderating Effect of Cognitive Capability on Task Conflict: A Longitudinal Study of Task Conflict and Team Performance in Student Software Development Teams," *AMCIS 2007 Proceedings*), p. 23.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS quarterly*), pp. 75-105.

Hinkin, T. R. 1998. "A Brief Tutorial on the Development of Measures for Use in Survey Questionnaires," *Organizational research methods* (1:1), pp. 104-121.

Holmström, J., and Sawyer, S. 2011. "Requirements Engineering Blinders: Exploring Information Systems Developers' Black-Boxing of the Emergent Character of Requirements," *European Journal of Information Systems* (20:1), pp. 34-47.

Inayat, I., and Salim, S. S. 2015. "A Framework to Study Requirements-Driven Collaboration among Agile Teams: Findings from Two Case Studies," *Computers in Human Behavior* (51, Part B), pp. 1367-1379.

Jehn, K. A. 1995. "A Multimethod Examination of the Benefits and Detriments of Intragroup Conflict," *Administrative science quarterly*), pp. 256-282.

Jehn, K. A. 1997. "A Qualitative Analysis of Conflict Types and Dimensions in Organizational Groups," *Administrative Science Quarterly* (42:3), pp. 530-557.

Jehn, K. A., and Mannix, E. A. 2001. "The Dynamic Nature of Conflict: A Longitudinal Study of Intragroup Conflict and Group Performance," *Academy of management journal* (44:2), pp. 238-251.

Jiang, P., and Fu, Y. 2009. "A New Conceptual Architecture to Enable Ipss as a Key for Service-Oriented Manufacturing Executive Systems," *International Journal of Internet Manufacturing and Services* (2:1), pp. 30-42.

Johnson, R. B., and Onwuegbuzie, A. J. 2004. "Mixed Methods Research: A Research Paradigm Whose Time Has Come," *Educational researcher* (33:7), pp. 14-26.

Joslin, R., and Müller, R. 2016. "The Impact of Project Methodologies on Project Success in Different Project Environments," *International Journal of Managing Projects in Business* (9:2), pp. 364-388.

Khan, S. U., Niazi, M., and Ahmad, R. 2012. "Empirical Investigation of Success Factors for Offshore Software Development Outsourcing Vendors," *IET Software* (6:1), pp. 1-15.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. 2009. "Systematic Literature Reviews in Software Engineering–a Systematic Literature Review," *Information and software technology* (51:1), pp. 7-15.

Kline, R. B. 2015. *Principles and Practice of Structural Equation Modeling*. Guilford publications.

Komoto, H., and Tomiyama, T. 2009. "Design of Competitive Maintenance Service for Durable and Capital Goods Using Life Cycle Simulation," *International Journal of Automation Technology* (3:1), pp. 63-70.

Levina, N. 2005. "Collaborating on Multiparty Information Systems Development Projects: A Collective Reflection-in-Action View," *Information Systems Research* (16:2), pp. 109-130.

Liang, T.-P., Jiang, J., Klein, G. S., and Liu, J. Y.-C. 2009. "Software Quality as Influenced by Informational Diversity, Task Conflict, and Learning in Project Teams," *IEEE Transactions on Engineering Management* (57:3), pp. 477-487.

Liang, T.-P., Jiang, J., Klein, G. S., and Liu, J. Y.-C. 2010. "Software Quality as Influenced by Informational Diversity, Task Conflict, and Learning in Project Teams," *IEEE Transactions on engineering management* (57:3), pp. 477-487.

Liu, C.-L. 2016. "Cdnfre: Conflict Detector in Non-Functional Requirement Evolution Based on Ontologies," *Computer Standards & Interfaces* (47), pp. 62-76.

Liu, J. Y.-C., Chen, H.-G., Chen, C. C., and Sheu, T. S. 2011. "Relationships among Interpersonal Conflict, Requirements Uncertainty, and Software Project Performance," *International Journal of Project Management* (29:5), pp. 547-556.

Lovelace, K., Shapiro, D. L., and Weingart, L. R. 2001. "Maximizing Cross-Functional New Product Teams' Innovativeness and Constraint Adherence: A Conflict Communications Perspective," *Academy of management journal* (44:4), pp. 779-793.

Macaulay, A. L. 1999. "Seven-Layer Model of the Role of the Facilitator in Requirements Engineering," *Requirements Engineering* (4:1), pp. 38-59.

Manzini, E., Vezzoli, C., and Clark, G. 2001. "Product-Service Systems: Using an Existing Concept as a New Approach to Sustainability," *Journal of Design Research* (1:2), pp. 27-40.

Marczak, S., and Damian, D. 2011. "How Interaction between Roles Shapes the Communication Structure in Requirements-Driven Collaboration," *Requirements Engineering Conference (RE), 2011 19th IEEE International*: IEEE, pp. 47-56.

Mastrogiacomo, L., Barravecchia, F., and Franceschini, F. 2019. "A Worldwide Survey on Manufacturing Servitization," *The International Journal of Advanced Manufacturing Technology* (103:9), pp. 3927-3942.

Maussang, N., Zwolinski, P., and Brissaud, D. 2009. "Product-Service System Design Methodology: From the Pss Architecture Design to the Products Specifications," *Journal of Engineering design* (20:4), pp. 349-366.

McDonald, R. P., and Ho, M.-H. R. 2002. "Principles and Practice in Reporting Structural Equation Analyses," *Psychological methods* (7:1), p. 64.

Meier, H., Roy, R., and Seliger, G. 2010. "Industrial Product-Service Systems—Ips2," *CIRP Annals-Manufacturing Technology* (59:2), pp. 607-627.

Mont, O. K. 2002. "Clarifying the Concept of Product–Service System," *Journal of cleaner production* (10:3), pp. 237-245.

Morelli, N. 2006. "Developing New Product Service Systems (Pss): Methodologies and Operational Tools," *Journal of Cleaner Production* (14:17), pp. 1495-1501.

Morgan, D. L. 2007. "Paradigms Lost and Pragmatism Regained: Methodological Implications of Combining Qualitative and Quantitative Methods," *Journal of mixed methods research* (1:1), pp. 48-76.

Myers, M. D., and Newman, M. 2007. "The Qualitative Interview in Is Research: Examining the Craft," *Information and organization* (17:1), pp. 2-26.

Paré, G., Trudel, M.-C., Jaana, M., and Kitsiou, S. 2015. "Synthesizing Information Systems Knowledge: A Typology of Literature Reviews," *Information & Management* (52:2), pp. 183-199.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. 2007. "A Design Science Research Methodology for Information Systems Research," *Journal of management information systems* (24:3), pp. 45-77.

Pernstål, J., Gorschek, T., Feldt, R., and Florén, D. 2015. "Requirements Communication and Balancing in Large-Scale Software-Intensive Product Development," *Information and Software Technology* (67), pp. 44-64.

Peruzzini, M., and Wiesner, S. 2019. "Emergence of Product-Service Systems," in *Systems Engineering in Research and Industrial Practice: Foundations, Developments and Challenges,* J. Stjepandić, N. Wognum and W. J. C. Verhagen (eds.). Cham: Springer International Publishing, pp. 209-232.

Putnam, L. L., and Poole, M. S. 1987. "Conflict and Negotiation,").

Raja, J. Z., and Frandsen, T. 2017. "Exploring Servitization in China," *International Journal of Operations & Production Management*).

Recker, J. 2012. *Scientific Research in Information Systems: A Beginner's Guide*. Springer Science & Business Media.

Reim, W., Parida, V., and Örtqvist, D. 2015. "Product–Service Systems (Pss) Business Models and Tactics – a Systematic Literature Review," *Journal of Cleaner Production* (97), pp. 61-75.

Rizzo, J. R., House, R. J., and Lirtzman, S. I. 1970. "Role Conflict and Ambiguity in Complex Organizations," *Administrative science quarterly*), pp. 150-163.

Rosenkranz, C., Charaf, C. M., and Holten, R. 2013. "Language Quality in Requirements Development: Tracing Communication in the Process of Information Systems Development," *Journal of Information Technology* (28:3), pp. 198-223.

Rowe, F. 2014. "What Literature Review Is Not: Diversity, Boundaries and Recommendations." Taylor & Francis.

Runeson, P., and Höst, M. 2009. "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical software engineering* (14:2), p. 131.

Schultze, U., and Avital, M. 2011. "Designing Interviews to Generate Rich Data for Information Systems Research," *Information and organization* (21:1), pp. 1-16.

Seregni, M., Sassanelli, C., Cerri, D., Zanetti, C., and Terzi, S. 2016. "The Impact of Iot Technologies on Product-Oriented Pss: The "Home Delivery" Service Case," *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*: IEEE, pp. 1-5.

Shameem, M., Chandra, B., Kumar, C., and Khan, A. A. 2018. "Understanding the Relationships between Requirements Uncertainty and Nature of Conflicts: A Study of Software Development Team Effectiveness," *Arabian Journal for Science and Engineering* (43:12), pp. 8223-8238.

Shih, L.-H., Lee, Y.-T., and Huarng, F. 2016. "Creating Customer Value for Product Service Systems by Incorporating Internet of Things Technology," *Sustainability* (8:12), p. 1217.

Shimomura, Y., and Hara, T. 2010. "Method for Supporting Conflict Resolution for Efficient Pss Development," *CIRP annals* (59:1), pp. 191-194.

Simons, T. L., and Peterson, R. S. 2000. "Task Conflict and Relationship Conflict in Top Management Teams: The Pivotal Role of Intragroup Trust," *Journal of applied psychology* (85:1), p. 102.

Sommerville, I., and Sawyer, P. 1997. "Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering," *Annals of software engineering* (3:1), pp. 101-130.

Song, W. 2017. "Requirement Management for Product-Service Systems: Status Review and Future Trends," *Computers in Industry* (85), pp. 11-22.

Song, W., and Sakao, T. 2016. "Service Conflict Identification and Resolution for Design of Product–Service Offerings," *Computers & Industrial Engineering* (98), pp. 91-101.

Spanoudakis, G., and Zisman, A. 2001. "Inconsistency Management in Software Engineering: Survey and Open Research Issues," in *Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals*. World Scientific, pp. 329-380.

Tashakkori, A., Teddlie, C., and Teddlie, C. B. 1998. *Mixed Methodology: Combining Qualitative and Quantitative Approaches*. Sage.

Teddlie, C., and Tashakkori, A. 2010. "Overview of Contemporary Issues in Mixed Methods Research," *Handbook of mixed methods in social and behavioral research* (2), pp. 1-41.

Tsai, K.-H., and Hsu, T. T. 2014. "Cross-Functional Collaboration, Competitive Intensity, Knowledge Integration Mechanisms, and New Product Performance: A Mediated Moderation Model," *Industrial Marketing Management* (43:2), pp. 293-303.

Tukker, A. 2004. "Eight Types of Product–Service System: Eight Ways to Sustainability? Experiences from Suspronet," *Business strategy and the environment* (13:4), pp. 246-260.

Vasantha, G. V. A., Roy, R., Lelah, A., and Brissaud, D. 2012. "A Review of Product–Service Systems Design Methodologies," *Journal of Engineering Design* (23:9), pp. 635-659.

Venable, J. 2006. "A Framework for Design Science Research Activities," *Emerging Trends and Challenges in Information Technology Management: Proceedings of the 2006 Information Resource Management Association Conference*: Idea Group Publishing, pp. 184-187.

Wall Jr, J. A., and Callister, R. R. 1995. "Conflict and Its Management," *Journal of management* (21:3), pp. 515-558.

Webster, J., and Watson, R. T. 2002. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," *MIS quarterly*), pp. xiii-xxiii.

Wiesner, S., Marilungo, E., and Thoben, K.-D. 2017. "Cyber-Physical Product-Service Systems: Challenges for Requirements Engineering (Mini Special Issue on Smart Manufacturing)," *International journal of automation technology* (11:1), pp. 17-28.

Wolfenstetter, T., Basirati, M. R., Böhm, M., and 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).

Yang, L.-R., Chen, J.-H., and Wang, X.-L. 2015. "Assessing the Effect of Requirement Definition and Management on Performance Outcomes: Role of Interpersonal Conflict, Product Advantage and Project Type," *International Journal of Project Management* (33:1), pp. 67-80.

Yang, L., Xing, K., and Lee, S. 2010. "A New Conceptual Life Cycle Model for Result-Oriented Product-Service System Development," *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics*, pp. 23-28.

Yang, X., Moore, P., Pu, J.-S., and Wong, C.-B. 2009. "A Practical Methodology for Realizing Product Service Systems for Consumer Products," *Computers & Industrial Engineering* (56:1), pp. 224-235.

Yvonne Feilzer, M. 2010. "Doing Mixed Methods Research Pragmatically: Implications for the Rediscovery of Pragmatism as a Research Paradigm," *Journal of mixed methods research* (4:1), pp. 6-16.

Zacharewicz, G., Diallo, S., Ducq, Y., Agostinho, C., Jardim-Goncalves, R., Bazoun, H., Wang, Z., and Doumeingts, G. 2017. "Model-Based Approaches for Interoperability of Next Generation Enterprise Information Systems: State of the Art and Future Challenges," *Information Systems and e-Business Management* (15:2), pp. 229-256.

Zancul, E. d. S., Takey, S. M., Barquet, A. P. B., Kuwabara, L. H., Cauchick Miguel, P. A., and Rozenfeld, H. 2016. "Business Process Support for Iot Based Product-Service Systems (Pss)," *Business Process Management Journal* (22:2), pp. 305-323.

Zou, M., Basirati, M. R., Bauer, H., Kattner, N., Reinhart, G., Lindemann, U., Böhm, M., Krcmar, H., and Vogel-Heuser, B. 2019. "Facilitating Consistency of Business Model and Technical Models in Product-Service-Systems Development: An Ontology Approach," *IFAC-PapersOnLine* (52), pp. 1229-1235.

Appendix: Publications in Original Format

The papers will be presented with the following order.

- Main Publication: P3
- Main Publication: P4
- Not Included in Review and Evaluation: P1
- Not Included in Review and Evaluation: P2
- Not Included in Review and Evaluation: P5
- Not Included in Review and Evaluation: P6

Main Publication: P3

Information and Software Technology 126 (2020) 106331

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Understanding the relationship of conflict and success in software development projects

Mohammad R. Basirati[a,*], Marko Otasevic[a], Koushyar Rajavi[b], Markus Böhm[a], Helmut Krcmar[a]

[a] Technical University of Munich, Germany
[b] Scheller College of Business - Georgia Institute of Technology, Georgia

A R T I C L E   I N F O

Keywords:
Conflict
Software project success
Software development
Non-human-rooted conflict
Organization size
Team size

A B S T R A C T

Context: Software development incorporates numerous people with diverse expertise and expectations. This makes conflict a common phenomenon in software development. Besides human causes, many conflicts in software development root in the tools and processes. Moreover, the growing role of software in any type of system is increasing the heterogeneity in software projects. The number and variety of tools and processes are increasing. Nevertheless, the relationship between conflicts, particularly rooted in non-human elements, and software project success is still unclear.

Objective: We aim to understand the impact of conflict on the success of software development projects for different types of conflict and different environments. Particularly, we distinguish between human-rooted conflict (HRC) and non-human-rooted conflict (NHRC). Moreover, we investigate whether organization size and team size moderate the impact of conflict on software project success.

Methods: First, we conduct a survey and analyze it using structural equation modeling (SEM) to investigate any correlation between conflict and software project success. Second, we explore the reasons behind the relationship between conflict and software project success by conducting 13 semi-structured expert interviews.

Results: HRC is always a threat to software project success for any organization or team size. Based on the interviews, resolving an HRC is regularly problematic. On the other hand, NHRC is negatively correlated with software project success only in corporate organizations and small teams. High coordination overhead and dependency on tools and processes make NHRC more influential in corporate organizations. In contrast, overlooking non-human elements and lack of experienced individuals in smaller teams make them more vulnerable to NHRC.

Conclusion: While the detrimental impact of HRC is constant for software project success, NHRC can be controlled efficiently. Corporate organizations need to frequently improve the non-human elements in the development. Smaller teams should expect tools and processes to be significantly influential in their success.

1. Introduction

Although software development already comprises numerous cross-domain collaborations [16], recent paradigms such as product-software systems (PSS), cyber-physical systems (CPS) and internet-of-things (IoT) further increase the number as well as complexity of software projects. Because when developing these kinds of systems, higher number of teams and departments from heterogeneous domains are involved, which need to work jointly on interdependent requirements of the system [51,67], teams face higher number of differences between stakeholders' perspectives, domain knowledge and processes. Working in such cross-domain settings leads to growth in both the number and variety of conflicts, with which we have to deal [14,39,43,49,64]. Functional diversity among the team members is acknowledged as a source of conflict, since it influences the team's cognitive diversity and consequently, the level of conflicts during decision making [46,50]. There is evidence that merely a higher diversity in a system's requirements is directly associated with higher interpersonal conflicts [42].

Conflicts are mostly perceived as a negative matter that changes communications, behaviors and structures [66]. Particularly in cross-domain collaborations, conflict impedes realizing innovative solutions [39,64]. An empirical study shows that interpersonal conflicts, such as disagreement, interference and negative emotions, always have negative influence on software development, regardless of how they are managed [8]. Liang et al. [41] found that while task conflict could increase the team performance in a software project, value conflict decreased the performance. [59] analyzed the impact of task and relationship conflict

* Corresponding author.

https://doi.org/10.1016/j.infsof.2020.106331
Received 29 November 2019; Received in revised form 21 April 2020; Accepted 27 April 2020
Available online 21 May 2020

in software projects and the findings show that relationship conflict has significantly a higher negative influence in comparison to task conflict.

Thus, the literature confirms that different types of conflicts have dissimilar influence on project performance. Nevertheless, existing studies primarily investigated conflicts rooted in human factors such as value, relationship or people's roles and responsibilities. Based on a McKinsey consulting company [53] report, in actual practice, more than 70% of general negotiations conflicts fail due to non-content factors, which, in addition to people-related issues, consist of processes (e.g., agenda, schedule, location, etc.). Such importance of non-human factors is also reflected in other aspects of software development. For example, an empirical study by Seth et al. [58] shows that in addition to human factors, non-human factors such as tools, methods, and infrastructure distinctively play a vital role in the construction of quality into software. Furthermore, the growth of distributed software development intensifies the importance of non-human factors as tools and methods play a bigger role in facilitating the remote collaborations [54]. The distinction between human and non-human factors is also acknowledged by the studies in the field of organization research [63]. Because the way that human and non-human factors influence on the project may differ substantially. Moreover, it is expected that dissimilar solutions are employed to tackle the negative consequences of human and non-human factors such as the case of conflict resolution [63]. For example, the way we manage an error-prone tool is usually very different from dealing with a problematic employee. As in the former, we may easily change the tool, while replacing an employee is not a straightforward process.

The role of human factors in conflict management has been studied extensively in the literature. However, despite the importance of non-human related factors, past research has not systematically investigated their effect on software project success and as such, our knowledge regarding the relative importance of human vs. non-human conflicts is limited. Recognizing the different impacts of human-rooted conflict (HRC) and non-human-rooted conflict (NHRC) on software projects help us to assess the situations of a software project more precisely. We would be able to make better decisions in case of a conflict and plan for future improvements effectively.

For further clarification, we distinguish between conflicts rooted in human-factors and conflicts rooted in non-human factors. However, any conflict may be reflected in human behaviors and associated with human factors. In this study, we focus only on the root cause of conflict as a major determinant of a conflict's nature and impact. To increase our knowledge regarding the influence of both HRC and NHRC, we follow a mixed-method research approach (Venkatesh et al. [65]). First, we perform a quantitative analysis by conducting a survey study. The first part of our research shows whether there is a correlation between presence of conflict and lower or higher project success measures. We next aim to explore the reasons and ways in which HRC and NHRC affect software project success. To this end, we conduct expert interviews. The expert interviews also provide interpretations of findings from the survey. Our research aims to answer the following conducting research question:

"How do HRC and NHRC influence success of software projects?"

The survey is conducted using an online questionnaire distributed among active professionals in software development projects. We use structural equation modeling techniques (SEM) [6,37] to analyze the survey. To gain more insights into how conflicts affect software project success, we conducted 13 semi-structured expert interviews [20].

The results of this study assert that there is a negative correlation between both HRC and NHRC, and the success of software projects. Nevertheless, the size of an organization, i.e., corporate versus small and medium-sized enterprises (SME), determines the extent to which NHRC is influential. While in corporate organizations, NHRC is significantly correlated with lack of success, we do not find statistically significant effect of NHRC on software project success in SMEs. Furthermore, team size also moderates the effect of NHRC on software project success. Surprisingly, we found that smaller teams (consisting of fewer than 10 members) suffer more from NHRC, whereas in large teams (consisting of more than 10 members), there is no correlation between NHRC and software project success. Regarding HRC, we found it negative for software project success in all investigated situations. However, HRCs are less influential on software project success in small teams.

The findings of this study increase insights of managers into software projects regarding the importance of non-human elements, which can impede the success of the project. In addition, we show in which settings (based on team size and organization size), non-human elements gain more importance. Particularly, we exposed that in contrast to regular expectations, smaller teams neglect importance of non-human factors and suffer more from NHRC. Furthermore, we contribute to the literature by confirming that HRC is negatively correlated with the project success independently from the size of the organization or team.

2. Theoretical background and hypotheses

In this section, we clarify the concept of conflict and elaborate relevant classifications of conflict types. We introduce the definition of conflict for this study and outline what we mean by HRC and NHRC. Moreover, this section reviews the existing software project success factors, from which we have designed the survey. Finally, based on the findings from prior research, we build four hypotheses for this study.

2.1. Conflict

Conflict is a broad concept and in many cases is used interchangeably with other notions such as dispute, disagreement or inconsistency based on context and situation. While it is easy to recognize a conflict situation, it is hard to define conflict as a concept [15]. The Oxford English Dictionary describes conflict in such terms as "serious disagreement," "opposing feelings or needs," "serious incompatibility" and "being incompatible" or "at variance." Easterbrook et al. [15] define conflict as the interaction of interdependent parties whose goals oppose and interfere with each other, emphasizing interaction, interdependence and incompatible goals. Conflict is defined by Wall and Callister [66] as "a process in which one party perceives that its interests are being opposed or negatively affected by another party."

For the purpose of this study and sake of consistency throughout the paper, we introduce a new definition of conflict, which does not refute existing definitions, but rather is more comprehensive. Based on existing definitions, we extracted three principles that exist for every conflict or inconsistency. First, two or more elements are involved. Second, there is a relationship between the involved elements. Third, the state of elements and their relationships are different to defined objectives of the involved elements. Therefore, based on these three principles, in this study, we define a conflict as *an undesired variance between two or more related elements*. We believe that this definition is more inclusive while complying with previous definitions.

Moreover, the literature highlights the multidimensional nature of conflict and accordingly introduces different types of conflicts. Relationship conflict type represents for the interpersonal conflicts, which are mostly due to the differences in personalities or negative emotions among people. Task conflict reflects disagreements on which tasks should be implemented and the why those tasks should be accomplished. Task conflict can be caused by many different sources such as contradicting requirements, lack of resources and so on. Process conflicts show disagreements about the overall process during implementation of a task, in which responsibilities and resources are assigned [29]. While task conflict is more about the content of the tasks, process conflict reflects the conflicts in logistics, scheduling and so on [21]. Role conflict addresses incompatibility in requirements of a role, in which a conflict emerges between people and their roles. For example, a role might be incompatible with a person's capabilities, values, resources or even other roles [55]. There is almost an overlap between role ambiguity and role conflict [57].

Existing studies mostly address conflicts from a merely human and social perspective, while tools and methods play an increasingly significant role in software development. For example, DevOps' processes and tasks heavily depend on the involved tools with a growing number [35,38]. Moreover, the smooth progress of a software development project relies on knowledge sharing tools such as wikis and communication tools, particularly in the case of distributed development settings [69]. In general, with increase in cross-domain and cross-site development, we face higher number of conflicts that arise from technical and non-human factors. This is because work settings contain heterogeneous elements that are originally suited to a domain or development site. For example, the number and variety of tools for development is rising. Incompatible and inappropriate tools lead to loss of information, which consequently may create a conflicting situation. Similarly, work processes and methodologies have a huge impact on the success of a software project [2]. Work processes or methodologies used by different teams may be conflicting with each other. For instance, a team that uses the waterfall approach has to collaborate with a team that is flexible in their approach. The way these two teams communicate and understand each other is heavily influenced by the differences between their overall work processes. Therefore, such a difference can easily trigger a new conflicting situation. Furthermore, a considerable amount of development knowledge such as business needs, system requirements and progress reports are communicated via documents that are not up-to-date or contain inconsistent information. Conflicting contents and information can be easily escalated among people.

To address this issue, we employ a socio-technical system (STS) theory lens. According to STS, two jointly independent, but correlative interacting systems form a work system composed of the social and the technical aspects. The technical system reflects the processes and technologies and the social system reflects individual's attributes and their relationship with the other individuals. Based on STS theory, both the social and the technical systems jointly create the output for a work system [10]. Thus, based on STS theory, software development is affected by both human factors, i.e., the social system, and non-human factors, i.e., the technical system. Accordingly, we introduce this distinction between the conflicts caused by human factors and the conflicts caused by non-human factors. Such a distinction acknowledges the importance of non-human factors and clarifies their role in conflicts and their impact on software project success. Furthermore, it enables us to choose effective strategies that prevent more conflicts or resolve existing ones. Therefore, we propose human-rooted conflict (HRC) and non-human-rooted conflict (NHRC) as two new types of conflict, which we analyze in this study. We formally define **HRC** *as a conflict that is rooted essentially in human factors, which are related to the general interests or background of a person such as personality or culture. In contrast,* ***NHRC*** *is a conflict that is exclusively rooted in non-human factors such as tools, processes or artifacts.* Table 1 provides several concrete examples of HRC and NHRC.

To clarify more, the classification of conflicts to HRC and NHRC focuses only on the roots of conflicts, excluding any other aspect of conflicts. Any conflict, whether, HRC or NHRC, could be reflected in human behaviors and be escalated to high-level social interactions. Therefore, what distinguishes an NHRC from HRC, is not whether people are involved in the conflict or not, but whether the root cause of conflict is a human factor or a non-human factor.

Such a new classification enables us to investigate precisely the impact of growing conflicts due to non-human factors on software project success. For example, we distinguish between a conflict caused by contradictory interests or personalities and a conflict caused by different tools, work processes and methodologies. Inconsistencies in artifacts such as requirements documents or legacy codes are another example of NHRC. Not only such inconsistencies can be considered as a conflict based on the conflict definition of this study, but also, they can be cause for more conflicts in higher levels. An inconsistency in defined software requirements can lead to incorrect implementation, delayed deliveries and consequently more conflicts [70]. Therefore, this study investigates existence of inconsistencies and errors in artifacts as an indicator for NHRC. Nonetheless, previous differentiations between conflict types could not cover NHRC sufficiently. As shown in Table 2, the common analyzed conflict types in the literature mostly address HRC.

Table 1
Examples of HRC and NHRC.

HRC	NHRC
A conflict between stakeholders' priorities	A conflict between DevOps' tools and the real needs of the DevOps process
A conflict between two teams from different organizations (e.g. in an outsourcing scenario) caused by different organizational cultures	A conflict between tools used by different teams from different organizations or departments
A conflict between two teams from different departments caused by a difference in terminologies	A conflict between methodologies of two teams such as agile and waterfall
A conflict caused by challenging or incompatible personalities	A conflict between two work processes e.g. by blocking each other
A conflict between individuals due to different level of expertise	Inconsistencies between legacy code and the new code libraries

Table 2
Existing Conflict Types and HRC/NHRC – Full Circle presents full coverage; half-full circle presents half-coverage; empty circle presents no coverage.

	HRC	NHRC
Relationship Conflict	●	○
Task Conflict	●	◐
Process Conflict	◐	◐
Role Conflict	●	○

2.2. Software project success factors

Software projects succeed or fail in many different ways and it is oversimplifying to determine the success of a software project with few factors [44]. Most of the time, success is a vague concept and it is not clear what success is and when a project is successful [1,9,33]. Moreover, different stakeholders may have different opinions regarding the success of a project and we can only measure perceived success [27]. Hence, similar to measuring concepts such as trust, success should be measured indirectly using a construct [44].

Nevertheless, research strived for identifying factors that can indicate the success of a project. The classical measurement is "iron triangle" that consists of time, cost and quality. Many scholars have criticized the limited capability of the triangle factors particularly with regard to ambiguity of quality dimension [27]. Researchers added factors that are more specific to the main indicators of success, among which realization of organizational objectives and stakeholders' satisfaction are stated more frequently [5,7,27,31]. However, time and particularly cost factors are repeated in most of later studies and confirmed by empirical studies such as [36]. Furthermore, some studies only focused on success of a particular type of IS and presented success factors that cannot necessarily be generalized on any software project. For instance, accessibility and extensibility are counted as success criteria for software systems that support senior executives [52].

Considering multidimensionality and ambiguity of success and the limited capability of existing general success criteria, we use time, cost and stakeholder satisfaction as the most commonly used and approved criteria in the literature. Although these indicators cannot elaborate all relevant dimensions of the project success, the literature assert that, they are mostly valid for any type of software project. Hence, we employ these factors to examine an overall software project success construct. Moreover, to further clarify, it is worth mentioning that we aim to measure success of a software project and not success of a software. The extent to which a software project is successful does not necessarily reflect the success of resulting software system during its use [44].

2.3. Relationship between conflict and success

Findings of a series of studies on the relationship between conflict types and team performance showed that regardless of different situations, any conflict decreases the individuals' satisfaction [28,29]. Another study by Giebels and Janssen [19] investigated the relation between conflict and well-being of employees based on factors such as stress and exhaustion. The study showed that conflict, whether a task conflict or relationship conflict, is associated with reduced well-being; however, the conflict resolution style can moderate the effect of conflict.

In particular, relationship conflict is found to have negative consequences regardless of situations. Jehn and Mannix [30] found in a longitudinal analysis that successful groups had low levels of relationship conflict. Liang et al. [40] showed that in software development teams, relationship conflict generally has negative influences. This is confirmed by study of [22], which showed significant negative correlation between the relationship conflict and the success of a project. Not only do all the relevant studies show that relationship conflict has negative effects on team or project attributes, but [60] also pointed out that relationship conflict can exacerbate the negative effects of other types of conflict, such as task conflict. Furthermore, constructive conflict resolution and trust have been found positively correlated with high-level performance [7].

Similarly, role conflict is also found to be negative in most cases. Based on a comprehensive literature review of role conflict studies, it is found that role conflict is significantly related to depression among employees [57]. Another study confirmed that the higher the role conflict, the lower the level of job performance [17]. Mohr and Puck [45] found that managers in joint ventures who experienced high level of role conflict had a lower level of job satisfaction and a higher stress level. A more recent study showed that role conflict has a direct negative impact on employee performance. However, it could inspire employees' ideas for improvement [56].

Fewer studies investigated process conflict. Nevertheless, most of the findings showed that process conflict has negative effects on performance [21]. Jehn and Mannix [30] also observed that successful teams had low level of process conflict. Guang-dong [22] found that there is a significant negative correlation between process conflict and the success of project.

Nevertheless, the findings regarding task conflict are not as clear as relationship conflict and role conflict. It was found that team performance can be associated both positively and negatively with task conflict, depending on the group structure and type of the task [28,29]. For example, task conflict could improve the performance of non-routine tasks [28]. Also, the longitudinal analysis of [30] showed that moderate levels of task conflict exist in successful teams. Liang et al. [40] found that in software development teams, task conflict could lead to learning among team members. Their investigation into the relationship between task conflict and software quality showed no significant correlation. Moreover, Guang-dong [22] found significant positive correlation between task conflict and project success because it was able to increase the communication and trust among the team members. In contrast, He [24] found that in software development teams, task conflict is negatively associated with team performance and it worsens over time, although a team's cognitive capability could moderate the severity of such an impact. Hjerto and Kuvaas [25] differentiated between cognitive task conflict and emotional task conflict. They found that cognitive task conflict is negatively related to team performance, while emotional task conflict is positively related to task conflict.

Most of the conflict-related studies have focused on the relationship between conflict and group/team performance or efficiency and the influence on the project success is slightly covered. Nevertheless, there is greater supporting evidence of negative impact of HRC on the project attributes than any positive impact. Therefore, we frame our hypothesis regarding HRC as shown below:

H1: HRC is negatively correlated with software project success

Relatively speaking, there are many fewer studies with respect to NHRC. The overall findings, explained above, show that the more conflicts are related to human aspects (relationship conflict and role conflict), the more they negatively influence the team and project performance. Although process conflict is partly related to non-human factors, in most cases, it has been found to be negatively correlated with team and project performance. In contrast, task conflict as another conflict type partially related to NHRC has been found to have both negative and positive impacts on team and project success.

Furthermore, conflicts among requirements artefacts as a particular example of NHRC are recognized by researchers as one of the main failure reasons in software projects [3]. Nuseibeh et al. [48] argued that despite severe negative consequences of conflicts in requirements artefacts, we should have some degree of tolerance for such conflicts in order to increase the performance of software development in terms of speed and goal accomplishment. Hadar and Zamansky [23] showed that people tend to not take responsibility for emerging conflicts in requirements artefacts because there is always a negative attitude towards them. Hence, such conflicts may escalate to interpersonal conflicts. In an empirical investigation, Yang et al. [68] revealed that there is a relationship between work processes and stability of requirements and project performance. Shameem et al. [59] found that in software projects, there is a relationship between variability of requirements artefacts and relationship conflict, which is strongly associated with reduced team effectiveness.

With regard to other elements such as methodologies and work processes, existing evidence shows a strong relationship between elements of a project methodology and the project success [32,36] also found that infrastructural elements have an important role in success of outsourcing projects.

To summarize, there is greater evidence for negative consequences of NHRC than potential positive ones. Thus, we define our hypothesis regarding NHRC as shown below:

H2: NHRC is negatively correlated with software project success

In addition, most of the existing conflict studies—including the reviewed studies in this section—show that settings in which conflict emerge is a decisive factor for consequences of a conflict. Therefore, to dive deeper into the situations in which negative effects of NHRC can be worse, moderated or even become positive, we also outline two more hypotheses about moderating effects of team size and organization size on NHR conflicts.

Team size is a significant factor that can affect a team's outcome [62]. For example, larger teams usually experience a higher number of task conflicts [4], however, they benefit from the more formal defined structures [18]. Moreover, larger teams typically have more social factions and less cohesion, which may intensify tensions [46]. Various studies have shown the moderating effect of team size in conflict analysis. For example, the study of [25] shows that the level of task conflict is less influential in large teams, whereas it is more influential in small teams. George et al. [18] also found that team size moderates the relationship between task conflict and process conflict. We also suggest that team size moderates the relationship of NHRC and software project success. In particular, as teams grow, more processes, tools and documentations are employed. In contrast, smaller teams tend to manage issues in person and non-human factors are less involved. Consequently, the way

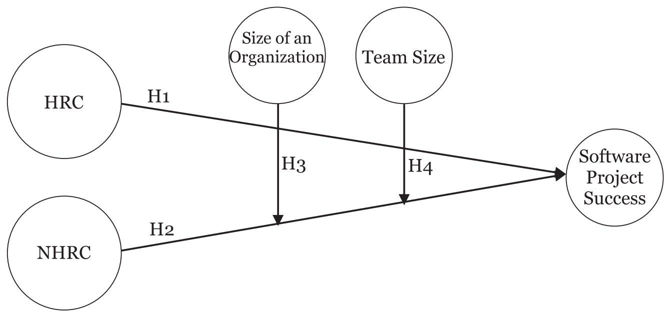


Fig. 1. Research model.

NHRC influences software project success in large teams is expected to be different in small teams.

In addition to team size, we also suggest that the size of an organization moderates the relationship between NHRC and software project success. As organizations grow, they develop structures, processes and culture. Such characteristics of an organization are influential on decision-making processes and how conflicts are experienced [46]. Particularly with regard to software projects, the effects of organizational factors on the outcome of a software project vary largely, depending on the size of the organization [34]. Furthermore, analyzing the effect of an organizational contextual factor—e.g., organization size—in addition to team size provides a deeper view into contextual factors that determine the relationship between conflict and software project success, especially since the dependent variable of this study—software project success—stands more on an organizational level than group level. Thus, we frame the fourth hypothesis regarding moderating effect of organization size on the correlation between NHRC and software project success.

H3: The size of an organization moderates the relationship between NHRC and software project Success

H4: Team size moderates the relationship between NHRC and software project Success

3. Research design

We conducted mixed-method research to answer the research question of this study and gain more insights regarding the research model (see Fig. 1). The mixed-method approach allows us to develop a deeper understanding of the phenomenon of interest [65]. We first explore the conflict problem using the survey study. We will then aim to find explanations of what we observed by conducting expert interviews. Hence, the purpose of our mixed-method research is “expansion,” based on which we expand and explain the findings of the former study by the later study [65]. In the following section, we will first describe data collection and design of measurements for the survey. We will then elaborate on the interview process.

3.1. Survey: data collection

We collected the data using an online questionnaire. The target group of the survey was professionals who work in software development projects, whether in managerial roles or technical development roles. No probabilistic method was used, since we did not focus on any particular type of respondent. Distributing the questionnaire started on 30 January 2019 and ended 1 April 2019.

We distributed the questionnaire among existing contacts, industrial partners in research projects and via LinkedIn. The industry contacts were also asked to distribute the survey among their relevant internal teams. Regarding LinkedIn, we contacted people in two ways. First, we identified people who were eligible for filling out the questionnaire. Then through personal messages, we asked them to fill out the questionnaire. We also posted the questionnaire to LinkedIn groups, in which software development professionals communicate on a particular software development topic. To avoid any misunderstanding, we explicitly emphasized in every message or LinkedIn post that only active persons in software projects are eligible to fill out the questionnaire. We also emphasized this requirement one more time at the beginning of the survey.

3.2. Survey: questionnaire and measurements

To design the questions, we used card sorting following the procedure of [47] in a group of two researchers and a professional who is a product owner in a software project at a corporate software development company. The procedure followed several rounds in which we identified the questions and the scale in which questions could be answered and clarification of stated concepts in the questions. In these iterations, we evaluated response options, relevancy and comprehensiveness of both questions and responses similarly to a pretesting with three people.

The final questionnaire consists of two sections and three subsections. The respondents were not necessarily able to recognize the subsections, but the sections were explicitly specified. We explicitly asked the respondents to fill out the questionnaire based on their experiences in only one project, one in which they were or are active. The first section was dedicated to general information about the respondent. General information contained a question regarding the size of organization in which the respondent was active. The choices offered were corporate or SME. In addition, we asked about respondent’s position in the company and the years of experience. For both questions, we provided five predefined answers. Answers to the position question could vary from entry level to senior executive and executive manager. For each answer, we provided several examples as well to clarified them. Years of experience were broken down into classifications ranging from less than two years to more than twenty years. In the general information section, we also asked about the size of team in which the respondents were working. The respondents could select answers ranging from fewer than five, ten, twenty and more than twenty team members. Finally, we inquired about their country of residence.

The second section contained subsections involving their perceived success, the perceived presence/absence of HRC and the perceived presence/absence of NHRC (all based on the last project in which the respondent was active). The questions of the second section are presented in Table 3. To collect more responses, we aimed for a questionnaire that takes a few minutes and it is very easy to understand. Therefore, we used the minimum number of items for measuring each construct. Moreover, we designed the questions enough comprehensive in order to cover most aspects of the measured variable.

Five questions measured success (Q1 to Q5). We focused on the perceived success of the project based on the existing aspects discussed in the section of 2.2 Software Project Success Factors. The five questions addressed the speed at which changes were applied, meeting deadlines, meeting budget limits, meeting project goals and stakeholder satisfaction. There was one question regarding each aspect and the respondents could answer it based on a five-degree Likert scale.

Four questions addressed HRC (Q6 to Q9) and three questions addressed NHRC (Q10 to Q12). Regarding HRC, we mixed direct and indirect questions that could reveal the presence of conflict during a project. For example, Q8 indicates whether people had trouble due to variance in domain knowledge among the involved persons. In contrast, Q9 asked whether the respondent had experienced any conflict with other team members. We identified NHRC based on three aspects (discussed in Section 2.1). First, we examined those conflicts due to differences in work processes and methodologies. Second, we asked about appropriateness of the tools. Finally, we questioned presence of mistakes and inconsistencies in development artefacts such as documents. It is important to note that based on the formal definition of HRC and NHRC in Section 2.1, we distinguish between an instance of conflict and its root. While an instance of conflict is concrete and very specific, the root of a

Table 3
Second section of questionnaire.

	Software Project Success
Q1	How quickly does your team adjust to changing priorities?
Q2	How often does your team meet its deadlines?
Q3	How often do your projects go over their allocated budget/headcount?
Q4	How much of the projects' goals does your team meet?
Q5	How often are stakeholders (users, customers, management board, etc.) satisfied with the projects' results?
	Human-rooted Conflict
Q6	Members of this team admit mistakes, apologize, and share learnings with one another.
Q7	There are often tensions and conflicts in the room that do NOT get surfaced or resolved.
Q8	How often do you experience conflicting situations due to people from different domains and functions in the projects?
Q9	How often do your general interests and priorities conflict with the others during projects?
	Non-human-rooted Conflict
Q10	How efficiently do the methodologies (agile, waterfall, etc.) and processes (general workflow process) match with goals or people skills?
Q11	How appropriate do you find the tools (e.g. domain-specific tools, communication tools, etc.) that you and your team are using in workflow processes?
Q12	Are the major documents and product parts generally well-constructed, up-to-date and free of inconsistencies?

Table 4
Details of expert interviews.

ID	Organization Size	Position	Experience (Years)
I1	Corporate	Developer	3
I2	Corporate	Tester	1
I3	Corporate	Product Owner	4
I4	SME	Executive Manager	7
I5	Corporate	Middle Manager	6
I6	Corporate	Middle Manager	7
I7	SME	Developer	3
I8	Corporate	Developer	2
I9	Corporate	Scrum Master	4
I10	Corporate	Tester	2
I11	SME	Developer	6
I12	SME	Developer	3
I13	SME	Program Manager	4

conflict can be a very generic issue such as inappropriateness of tools or processes. Hence, the items rather aim to tackle the roots of conflicts, not specific instances.

3.3. Expert interview

The purpose of the expert interviews was to find explanations for the observations based on the survey study. Survey respondents who were interested in the study could provide their contact for further steps. We received 13 contacts who collaborated with us in conducting an interview. We performed semi-structured, in-depth expert interviews [20]. The interview guidelines consist of 5 primary leading questions, which were followed by 11 sub-questions. The first leading question addressed the general overview of the interviewee on conflict, such as personal definition, expected consequences and so on. The second and third leading questions focused on the relationship between conflict and software project success factors. The fourth and fifth leading questions investigated the effect of team size and the organization size on impact of conflict. The interviews took from 30 min to 1 h The interviews were transcribed and coded using an open-coding style. We searched for explanations and reasons that could be used to interpret the findings of the survey study. The details of the interviews are presented in Table 4.

4. Results

This section gives an overview of respondents' demographic profile, evaluation of the model and the results of hypotheses assessment and the interviews.

Table 5
Demographic Profile of Respondents.

Variable	Category	N	% of Respondents
Organization Type	Corporate	58	54.3
	SME	49	45.7
Position	Executive Manager	7	6.5
	Senior Manager	6	5.6
	Middle Manager	25	23.4
	Experienced	43	40.2
	Entry Level	26	24.3
Years of Experience	20+	12	11.2
	10 – 20	9	8.4
	5 – 10	22	20.6
	2 - 5	43	40.2
	0 – 2	21	19.6
Team Size	20+	19	17.8
	10 – 20	27	25.2
	5 – 10	38	35.5
	1 – 5	23	21.5
Region	West Europe	59	55.1
	East Europe	13	12.1
	Asia	6	5.7
	North America	27	25.2
	Rest of World	2	1.9

4.1. Demographic profile of respondents

We summarized the demographic profile of the respondents in Table 5. We received a total number of 107 responses. The number of responses from corporate firms are almost the same as the number of responses from SMEs. Regarding the position of the respondents in the organization, we have more responses from employees in the lower levels of organization. This is as we expected, since it was challenging to reach executive and senior managers to fill out the questionnaire. About 60% of the respondents have up to five years of experience and about 20% of the respondents have more than 10 years of experience. Hence, we expect the results to be representative of less experienced persons. The number of responses from different sizes of the teams are evenly divided and almost no group has a significantly higher number of responses. Based on the contacts that we had, most of the responses are from Europe and the second are a group from North America.

4.2. Measurement model assessment

Before studying the relationships between HRC, NHRC and software project success, several empirical issues must be examined. First, we must analyze whether the items that we have used to measure HRC and NHRC are in fact measuring two different constructs (rather than a single conflict construct). We use principal factor analysis [11] to determine

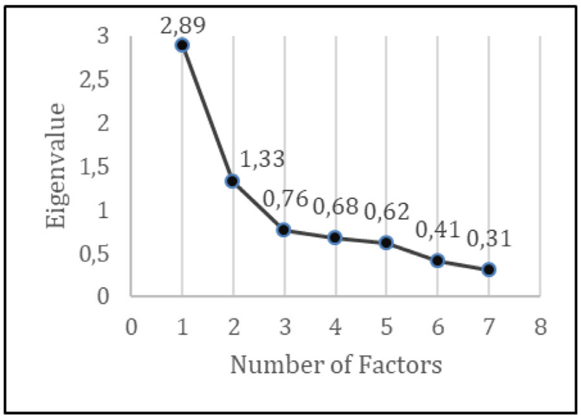


Fig. 2. Scree plot.

the optimal number of underlying factors that have been measured using the seven items that we used for measuring HRC and NHRC. To this end, we employ two established methods. First, we investigate Eigenvalues above 1 and second, we perform a Scree Test.

The method based on Eigenvalues retains any factor that has Eigenvalue of at least with the rationale being that any additional factor with a less than one Eigenvalue would be unstable. In our analysis, only two factors had Eigenvalues above 1 (2.89 and 1.33 respectively; a potential third factor had Eigenvalue of 0.76). Scree test was introduced by Cattell [12]. In this approach, the number of factors is determined based on the last Scree (sudden drop) in the plot Eigenvalues. Accordingly, as shown in Fig. 2, the last big drop occurs between second and third factors, therefore we retain the second factor but not the third one.

Next, we must examine whether the items used to measure the two constructs load strongly on their respective factors, without loading strongly on the other factor. In other words, we analyze whether the questions designed for measuring HRC are actually measuring HRC and not NHRC instead. We used the Varimax rotation to examine the pattern of loadings. The results (presented in Table 6) show that the items used to measure HRC load strongly on factor 1 (loadings are between 0.61 and 0.88) and that they do not load strongly on the second factor (loading are between 0.09 and 0.24). Similarly, items used to measure NHRC load strongly on factor 2 (loading are between 0.75 and 0.76) and they do not load strongly on the first factor (loadings are between −0.08 and 0.28). These results confirm that the items used to measure HRC and NHRC are appropriately measuring separate constructs.

As such, both methods point to a two-factor solution which is consistent with our theory that HRC and NHRC are two different types of conflicts. The two factors explain more than 60 percent of the variation that the seven items contain. Once we retain two factors, we do a Varimax rotation after the factor analysis to see the relationship between each survey questionnaire and each of the two factors. We report these factor loadings in Table 6. The factor loadings allow us to assess the empirical relationship between questionnaire items with those expected by the researcher at the time of item creation and data collection. Ideally,

Table 6
Conflict measurement questions and confirmatory factor analysis.

Items	Factor 1 (HRC)	Factor 2 (NHRC)
Q6	.72	.16
Q7	.88	.10
Q8	.78	.09
Q9	.61	.24
Q10	.28	.76
Q11	.21	.75
Q12	−0.08	.76

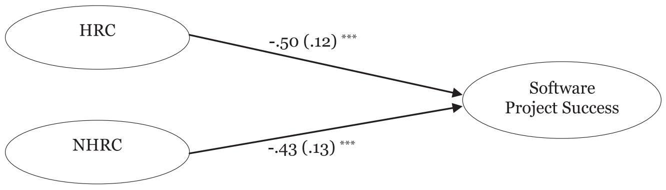


Fig. 3. Relation between HRC and NHRC, and Software Project Success (H1, H2).

one would want to see 1) related items to load strongly on the same factor (magnitude of loadings above 0.6) while loading only weakly on the other factor (magnitude of loadings below 0.4), and 2) related items load on the same factor in a similar direction (i.e., similar sign).

In our case, we expected Q6, Q7, Q8, and Q9 to strongly load on one factor (magnitudes of loadings above 0.6) while not strongly loading on the other factor (magnitude of loadings on factor 2 are all smaller than 0.3). The same thing can be said about the three other questions. Q10, Q11, and Q12 load strongly on the second factor (magnitude of loadings above 0.7) while loading weakly on the first factor (magnitude of loadings below 0.3). These loadings give us confidence that 1) Q6, Q7, Q8, and Q9 are measuring a similar concept, 2) Q10, Q11, and Q12 are measuring a similar concept, and 3) the concept that Q6, Q7, Q8, Q9 and the concept that Q10, Q11, Q12 are measuring are meaningfully distinct from each other – otherwise, the questionnaire items would have loaded strongly on both factors rather than only on one factor.

Finally, we examine Cronbach's reliability coefficient for each construct. Cronbach's reliability coefficient is 0.77 for the HRC suggesting a very good reliability. The reliability coefficient is 0.66 for NHRC suggesting an acceptable reliability. The reliability coefficient for the five items used to measure software project success is 0.69. Overall, the above tests provide empirical evidence for validity of the items used to measure HRC conflict, NHRC and software project success.

4.3. Hypothesis assessment

First, we present the results of SEM analysis[1] regarding first and second hypotheses (see Fig. 3). We examine the relationships between HRC, NHRC and software project success. The results suggest that HRC is negatively correlated with software project success (−0.50, p<.01), thereby confirming H1. In addition, we find that NHRC is negatively associated with software project success (−0.43, p<.01), thereby confirming H2.

We also evaluate the fitness of the model based on the existing approaches in SEM. The first approach is Root Mean Square Error of Approximation (RMSEA) that ranges from 0 to 1 with smaller values indicating less error (less discrepancy between data and hypothesized model). A value of 0.06 or less is an acceptable amount of RMSEA [26]. In our analysis, RMSEA is 0.04, which shows an acceptable fit for our model. Moreover, we calculated the Comparative Fit Index (CFI) that ranges from 0 to 1 with larger values indicating better fit between data and hypothesized model. The literature suggests that values above 0.95 indicate a decent fit [26]. The CFI for our study is 0.97 that again approves the fitness of our model. Finally, we calculated the Tucker-Lewis Index (TLI) that captures the difference between the chi-squared of the hypothesized model and the chi-squared of the null model. For TLI which also ranges between 0 and 1, higher values indicate better fit and values above 0.95 are desirable. In our analysis, the TLI is 0.96 that shows the proper fitness of our model.

In order to examine the hypothesis about organizational size (H3), we ran two separate SEM analyses; one for corporate organizations and one for SME organizations. We identified the size of an organization by asking the respondents directly. The results (see Figs. 4 and 5) suggest

[1] n = 107; *** p<.01; ** p<.05; * p<.10

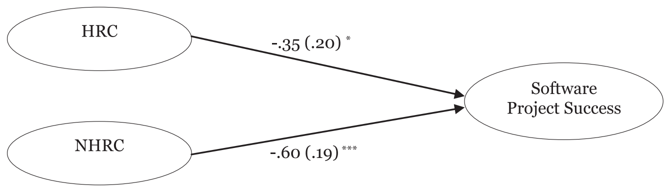


Fig. 4. HRC and NHRC, and Software Project Success Relation in Corporate Organizations ($N = 58$).

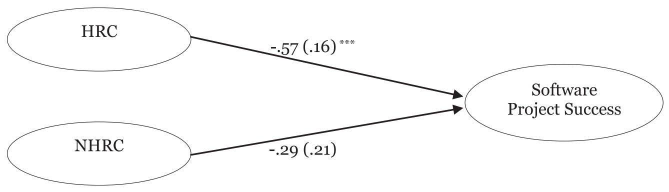


Fig. 5. HRC and NHRC, and Software Project Success Relation in SME Organizations ($N = 49$).

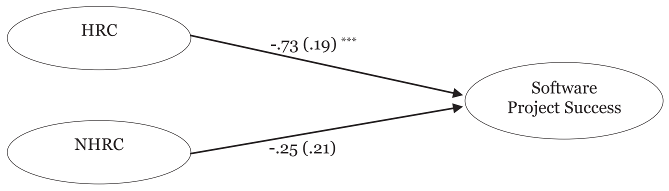


Fig. 6. HRC and NHRC, and Software Project Success Relationship in Large Teams (Team Size > 10, $N = 46$).

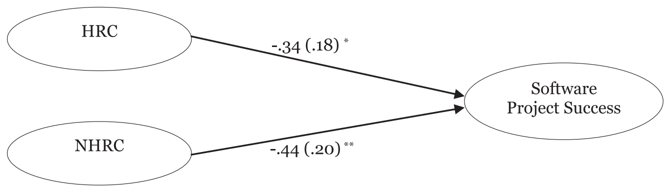


Fig. 7. HRC and NHRC, and Software Project Success Relation in Small Teams (Team Size <= 10, $N = 61$).

that NHRC and software project success are only significantly related in corporate organizations (−0.60, p<.01). We do not find a significant association between NHRC and software project success in SMEs (−0.29, p>.10). This confirms our third hypothesis (H3). Furthermore, we also analyzed the same moderating effect for HRC. The results show that there is a significant negative correlation between HRC and software project success in SMEs. Although such a correlation exists also in corporate organizations, it is less significant in terms of statistics and magnitude (−0.35, p<.10).

In order to examine the moderating role of team size (H4) we run two separate SEM analyses, one for small teams and one for large teams. We divide the responses into small and large teams based on the answer to the team size question. We chose 10 as the number of team members, based on which we classify the teams into small and large. Hence, small teams consist of 10 persons or fewer and large teams have more than 10 team members.

The results (see Figs. 6 and 7) suggest that NHRC and software project success are only significantly related in small groups (−0.44, p<.05). We do not find a significant association between NHRC and software project success in large groups (−0.25, p>.10). This confirms our third hypothesis (H4), that team size moderates the effect of NHRC on software project success. Moreover, we also examined moderating effect of team size on HRC and software project success. We find a strong relationship between HRC and software project success in large groups (−0.73, p<.01) whereas the relationship between HRC and software project success is relatively smaller in magnitude in smaller groups (−0.34, p<.10).

In addition, to make sure that the group size and organization size variables are reasonably distinct, we assessed the correlation between these two variables. Correlation of −0.18 ensures that the two variables are empirically unrelated.

4.4. *Interviewee explanations*

In general, the perspective of interviewees on conflict is very diverse, with attitudes ranging from assuring that conflict is mostly a negative phenomenon (I1, I4, I7, I8) to the arguments that conflicts should be celebrated (I5) and treated positively (I2, I3, I10, I12). Also, one interviewee argued that as long as differences and confrontations lead to a solution and do not block the progress, that they prefer not to call it a conflict (I11). Nevertheless, all interviewees reported that they experience conflict frequently in their everyday work.

Whether conflicts affect the project in the short or long term was mentioned by several interviewees. Three interviewees argued that in short term, all conflicts may have negative impact on the project because at a minimum, they hold up the project (I3, I12, I13). However, in the long run, people and organizations can learn from conflicts and if the impact is analyzed over the long term, it might be even positive. Particularly for NHRC, conflicts can show the deficiencies of the processes and tools, which are much easier to be adjusted and resolved (I12). In contrast, an interviewee reported that while the consequences of a conflict were obscure in short term, over the long term, several individuals changed their position in the company (I6). Furthermore, two interviewees discussed that the level, in which you work determine which types of conflict you experience more: for a low-level developer, it is mostly HRC, because non-human factors are already decided for them by managers (I11, I12).

The clearest impact of any conflict for interviewees was delay, which is true for almost any conflict (e.g., I1, I3, I5, I6, I13). Conflict, whether HRC or NHRC, means that things followed a path other than what was planned. Consequently, the schedule, priorities and objectives are affected and mostly in negative way (I4, I5, I13). In the following, we report the interviewees' experiences and opinions in detail on the ways HRC and NHRC affect project success. Moreover, we provide our findings from the interviews on the moderating role of organizational size and team size on conflict's impact on the software project success.

4.4.1. *How does HRC affect project success?*

The most experienced HRCs exist in working with an external person, either from another team or department or even from the same team, but who are in another location (I1, I2, I9, I11). Lack of awareness about the state of the other party, the motives and rationale behind the decisions and requests is a major factor in halting progress (I11, I13). External parties usually have different priorities and one party should wait for the other one to progress, which adds delay to the project (I1). This delay is in addition to regular delays caused by long rounds of discussions and meetings for resolving the conflicts. In such situations, conflict may also directly affect the quality and features of the final product. For instance, a development team may reject developing a feature requested by marketing team, simply because the motives are unclear, and they think the requested feature is absurd (I11, I13). Particularly, if the development team has already experienced the fact that their effort and time are dismissed, they then feel that they cannot trust external parties. Dismissing developers' opinions and effort demotivates them and lowers their performance, which in turn leads to more delays and a lower quality outcome (I4, I6, I7).

As mentioned by interviewees, company culture is a major decisive factor for the level of impact that conflicts can exert (I6, I10). Company culture determines how a conflict should be treated and solved and if handled incorrectly, the conflict can negatively influence the success of a project. For example, very similar conflicts reported by two interviewees had very different outcomes (I10, I12). As a common emerging conflict, urgent change requests from sales or marketing teams are conflicting with the priorities of the product owner and development team. In one company this conflict led to tension, frustration and mistrust among the parties (I12, I13), whereas in another company, it became the basis for improving the processes and communication with the clients (I10).

Conflict between roles and the importance of roles governing how a conflict is managed were also reported frequently. One interviewee urged that there should be a leader in the team who can make final decisions and bring closure to discussions (I7). Several interviewees stressed that parties in higher positions experience conflicts differently than the low-level development team (I1, I11, I12, I13). For example, the product owners see a broader view in comparison to software developers. Consequently, product owners can better handle conflicts and mitigate the negative impacts more easily (I1). In contrast, developers may not see the big picture, but since they are the main creators of the final product, they take pride in its implementation and techniques, which may make them less tolerant of others' views (I3). Also, introducing new roles to a project may create new conflicts (I3). In general, the differences in priorities for different roles is one of the most frequent causes of conflicts, such as those among a management team, the product owners, the project managers and the developer team (I10, I12, I13). Conflicts in priorities directly affect quality of the delivery and the delivery time.

The openness of individuals can mitigate conflict, whereas having a biased attitude usually worsens conflicts and stops the work progress, which in turn leads to delays and lowered productivity. For example, one interviewee reported that usually, experienced developers are very closed to others' opinions and do not readily accept input from younger developers, which causes delays and wrong implementations (I1). In contrast, in a similar situation, with an open and experienced developer, a young developer could learn principles and some implicit expertise, and the experienced one could learn new technologies and trends (I9).

It is noteworthy to mention that HRC is more frequent in highly international teams where many individuals are from different cultures and usually speak their second or third language at everyday work. This leads to unexpressed opinions and motives which harm everyone, reduces trust, increases conflicts and impedes resolution (I3, I11).

Interviewees also stated several ways that conflicts can be positive for a project. The most-often mentioned way was when a conflict situation became the basis for learning. In that way, differences among perspectives, mind-sets, skills, etc. become teaching materials for the involved parties. For example, product design would be more complete as different mindsets are exposed to various aspects of the product design. Another discussed learning example was between an experienced developer and a young developer who were not able to benefit from their conflicts to learn from each other (I9). Conflicts could be exploited as a means of exposing members to improvement points, particularly for workflow procedures and development approaches.

In general, in the worst-case scenario, HRC conflicts can lead to a blame game among all involved parties, demotivating everyone and damaging the esprit de corps (I12). The more personal a conflict gets, the harder it gets for it to be resolved, causing more negative impact on the project outcome (I3, I9, I13). The negative impact of HRC on a project's success were clear to all interviewees. Even those team members who thought the conflict should be treated positively mostly experienced the conflict as having a negative impact on the project, although some positive outcomes were also reported. The interviewees mostly believed that HRC is harder to resolve and more influential for success of the project than NHRC (I5, I6, I11, I12).

4.4.2. *How does NHRC affect project success?*

Interviewees were not familiar with distinguishing between HRC and NHRC. However, they approved that such a classification is valid. They mostly used to view conflict as human tension, whether the roots are human factors or non-human factors. Most interviewees saw non-human factors as auxiliary and non-essential, and low-level developers in particular did not pay much attention to NHRC because most non-human factors such as tools and processes are defined by managers and such factors are considered constants for the rest of the development team (I11, I12). Nevertheless, after clearly introducing the concepts of HRC and NHRC, the interviewees described many NHRCs.

In many conflicting situations, NHRC exists as much HRC. For example, in working with external teams, NHRC happens almost as much as HRC, since external parties may use different processes, tools, standards and formats (I1, I2).

Some decision-making processes can inherently create more conflict, such as having many hierarchy levels (I2, I4). The success of a project can suffer considerably due to conflicts during the decision-making processes, since it has direct impact on the progress of work and the final product (I2). Sometimes team structure must be changed in order to avoid NHRC (I3). However, change in the team shape may introduce new conflicts to the team and take some time for adjustment (I8).

One interviewee discussed a process that they follow for assuring quality of the final product and its direct impact on the success of project (I6). In such a process, even small inconsistencies in documents can stop the progress and cause delays (I6). Generally, long processes add delay to the project (I2), but following no methodology and process also drives everything to a personal level, which can create more HRC (I7). Another problematic situation is when individuals are involved in more than one project at a time. This creates conflicts in priorities between the two projects. The individual involved in two projects usually experiences delay in one or both projects, and the quality of her/his outcome is often poor (I8).

There was a consensus among most of the interviewees that tools are very important and influential in projects (I1, I2, I3, I4, I6, I8, I13). This is interesting since they had always asserted the importance of tools, but they usually did not consider their role in creating conflicts. The noted important tools for the interviewees can be divided into three categories of communication tools (e.g., Slack), issue tracking tools (e.g., Jira) and development tools (e.g. Git). Using appropriate tools increases work efficiency and when there is a conflicting situation caused by tool-related aspects, work efficiency drops considerably (I2). For example, the connection of tools with each other can facilitate the work (I4, I5), particularly build automation tools (I13). In contrast, legacy systems that must be adjusted introduce delay to the project (I8). Also, when there is an individual who is not familiar with a tool, or a new tool is introduced to the project, can significantly slow down the progress of a project (I10, I12). One interviewee reported that DevOps-related tools and processes have the most direct impact on the project. Any conflict in DevOps issues adds considerable delay to the delivery time (I6).

Nevertheless, several interviewees argued that despite negative impacts of NHRC, it can be resolved much more easily than HRC (I1, I6, I12).

4.4.3. *How does the size of an organization moderate the impact of a conflict on a project's outcome?*

The survey results showed that there is a significant negative correlation between NHRC and project success in corporate organizations, whereas such correlation does not exist for SME organizations. Also, we found that HRC is negatively correlated with project success in both types of organizations, but it is more statistically significant for SME organizations. We asked interviewees about their expectations on how organization size moderates the relationship of conflict and success. In the following, we provide the interviewees' answers and their interpretation of the survey's results.

The interviewees all expected the same results as found in the survey. They highlighted the importance of coordination in corporations due to large numbers of divisions, departments and teams. Such coordination is highly dependent on non-human factors such as tools, processes and artifacts (I3, I6, I11, I12, I13). For example, as discussed for HRC, transparency of involved parties' motives hugely affects how conflicts emerge and influence a project. In larger organizations, non-human factors such as processes and tools are vital for creating such transparency, which makes NHRC more influential (I3).

Another aspect of large organizations is that the tools and processes are mostly defined in a top-to-bottom manner (I6, I11, I12). Top managers determine non-human elements based on the average needs of the employees. There is always a team, whose needs differ from typical average needs. Hence, such top-to-bottom decision makings always lead to NHRC for low-level teams (I11). In contrast, SME organizations can customize processes and tools to their needs and employees, making them more efficient and less conflict prone (I6, I11). In addition, if an NHRC emerges in SME organizations, it is much easier to manage and resolve it (I6), whereas in large organizations, any small NHRC can have a domino effect and influence many teams and subprojects (I12).

4.4.5. How does team size moderate a conflict's impact on project outcome?

The survey results showed that there is a significant negative correlation between NHRC and project success in small teams, whereas such correlation does not exist in large teams. Similar to organization size, in both large and small teams, HRC was found to negatively correlate with project success. We asked the interviewees about their expectations on moderation effect of team size and their interpretation the survey's results.

In contrast to the survey's results, the interviewees expected that NHRC would be more influential in large teams instead of small teams. Their reasoning here was the same as for the importance of NHRC in larger organizations. In other words, the interviewees expected that more coordination in larger teams lead to the same consequences for NHRC impact on project success. Although the survey's results were surprising to them at first, they could relate and provide some hypotheses after the discussion advanced. In the following section, we provide the interpretation of the survey's results in two parts. First, we detail why in small teams NHRC was influential. Second, we elaborate the reasons for the neutrality of NHRC for large teams.

The smaller teams are, the more informal the handling issues and tasks (I1, I2, I6, I11, I12). In smaller teams, individuals tend to do tasks and manage issues verbally with more rounds of talks because it is possible and seems quicker (I4). Consequently, non-human factors such as processes are overlooked (I12, I13). Thus, in small teams, NHRC is more frequent and more challenging to resolve, which means it has a greater impact on a project's outcome.

Usually, it takes time for a team to grow. Hence, the deficiencies in a team's non-human factors team are mostly exposed and tackled as the team is growing (I13). Therefore, large teams use processes and tools that are established and their complications have already been resolved (I6, I13). Furthermore, in contrast to small teams, whose team members may be all inexperienced, large teams typically consist of experienced individuals as well as less-experienced developers (I1, I11, I13). Therefore, in the case of an NHRC, experienced team members manage the conflict quicker and more efficiently, preventing a major negative impact on project success. As reflected in discussions with an interviewee, lack of a leader in their small team led to resolution-less conflicts, which negatively impacted the project outcome (I7). Nevertheless, as can be seen from the survey results, such interventions from experienced experts may worsen the influence of HRCs.

Regarding the HRC, many interviewees (e.g., I1, I2, I3, I11) argued that in small teams, HRCs are resolved very quickly as communication is easy and team members know and trust each other. In large teams, there are always individuals who cannot get along with each other and high communication overhead among them hinders the resolution of such conflicts (I6, I13). Therefore, HRC would be more influential on project success in larger teams than in small teams.

5. Discussion

The growth of variety among domains and stakeholders in software development introduces higher number of conflicts in the projects, since diversity creates more sources of conflict [50,51]. Such a diversity is reflected in not only differences among people, but also in the variety of tools, methods and development mechanisms. In this study, we investigated how conflicts influence software project success, and whether the conflict originates from the non-human elements (NHRC) or human-related elements (HRC).

5.1. Discussing findings

We found that there are statistically significant negative correlations between both HRC and NHRC and software project success (p-value<0.01 for both HRC and NHRC). On the one hand, such results confirm the previously found negative consequences of conflict in the literature [22,24]. On the other hand, the findings question whether the positive effects of conflict such as learning and better ideation help software project success [13,40]. Based on the interviews, a possible explanation could be that conflict decreases software project success in short term. While in long term, conflict can be leveraged as an exposure to weaknesses and lessens-learned, which may increase the overall software project success. Several interviewees also argued that conflicts are not necessarily negative and should be treated as a positive issue more often. Nevertheless, the positive effects of conflicts may be limited to a particular scope and on a project level, they are either harmful or neutral. But we could not find any sign of positive effect of conflict on software project success in any situation.

One explanation is that conflict is a multidimensional phenomenon—it affects and is affected by various elements. Hence, narrowing down the variables of our research to only two types of conflict, software project success, team size and organization size, cannot present complex relation of conflict with other aspects of a software project. For example, as mentioned in several interviews, the conflict resolution style and attitude of those involved can determine negativity or positivity regarding a conflict's consequences. However, as the number of stakeholders and their conflicts grows, it becomes more challenging to deal with every single conflict in a constructive manner. Consequently, the overall impact of conflicts on software project success would be negative.

HRC was found to negatively correlate with software project success, regardless of team size and the size of an organization. In contrast, the results show that both team size and the size of an organization moderate the effect of NHRC. Corporate organizations have a high number of stakeholders and hierarchies that change over time. It would be challenging to update numerous stakeholders on goals and progress. The information may get lost, be rendered incomplete, and inconsistent information may lead to errors and delays. Moreover, corporations have a higher number of remote employees since development sites are scattered around the world. Hence, not only is the communication load inherently heavier, but also working in different sites and positions leaves no room for watercooler conversations. Therefore, people are more dependent on the defined processes, tools and documentations in order to keep everyone apprised and able to deal with different situations. Consequently, NHRC can significantly influence project outcome. In contrast, in SMEs, the problematic non-human factors can be managed quickly through short conversations and meetings. Therefore, findings show that the impact of NHRC would be dismissed in favor of the success of the project. Even spending too much effort on tools and processes in SMEs could impede success of the project.

Findings involving the effect of team size was interestingly unexpected. In contrast to the results of organization size moderation, NHRC is negatively correlated with software project success in small teams (team size$<=10$). We found no significant correlation between NHRC and software project success in large teams (team size>10). Many interviewees were surprised by our findings. They expected that the same circumstances, that make NHRC more influential in large organizations would also make NHRC more influential in large teams. Nevertheless, we found several explanations in the interviews such as overlooking the non-human elements in small teams, lack of experienced individuals in small teams, incremental improvement of non-human elements in larger teams as they grow in size. Our results are aligned with the findings of [25], who found that task conflict is less influential in large teams, and [18], who found that large teams benefit from formal processes. Furthermore, we complete the prior research by providing explanations from the interviews on the ways, which team size moderates conflict's impact on software project success.

5.2. Theoretical and practical implications

We contribute to the literature in three ways. First, we extend the existing types of conflict by introducing HRC and NHRC types. Although human and non-human elements are intertwined in software development, acknowledging the dissimilar nature of these two kinds of elements help us to better understand the dynamic of conflicts in software development. This distinction is particularly important for studying software projects, which deal with various domains and teams, because it is common practice that teams from different domains employ dissimilar sets of tools and processes [61,67]. Furthermore, prior studies mostly focused on group-level implications of conflict (e.g., team performance). How conflict's influences on project outcome, particularly in software projects, was missing. We examined this kind of impact and found that conflicts, whether HRC or NHRC, have detrimental effects on the success of software development on a project level. Hence, we contribute to the literature on conflict in software development and the literature on software project success. Establishing upon the findings of this study, future research can incorporate conflict as an indicator or influential factor for software project success. Finally, we deepened our knowledge of conflicts by analyzing the moderating effect of two contextual factors: team size and organization size. The findings proved that the moderating effect of both factors is relevant for NHRC, whereas team size and organization size cannot change impact of HRC significantly. The findings involving NHRC in small teams provide new evidence of team complexity. This extends our knowledge about the impacts of organization and team on the software project success.

The findings implicate several managerial concerns. First, practitioners should notice that the presence of HRC is bad for the success of software projects in any situation. They must strive for a constructive atmosphere in the project to transform negative effects of conflicts into positive ones. Based on the interviews, culture of a company has a big role in the way, the conflicts are resolved. Second, managers in corporate organizations need to pay more attention to non-human elements. They should expect and prepare for NHRCs in order to resolve them properly. We recommend that corporations continuously improve their non-human elements such as processes, tools and mechanisms for increasing awareness among employees. On the other hand, SMEs should not waste their time and budget on creating extensive workflow processes and providing advanced tools, nor should project managers of small teams underestimate the impact of non-human factors. Although it might not be a project manager's priority to address tools and processes in small teams, such elements need to be managed and adjusted properly to achieve a more successful software project. In addition, developers in small teams should not overlook the importance of non-human elements and prepare themselves for NHRCs.

5.3. Limitations

The findings of this study are limited to its definitions for HRC and NHRC. We provide pieces of evidence on how non-human factors can play a role in the success of software projects from a conflict lens. However, we cannot generalize the findings to various aspects of non-human factors in software development and project success. Moreover, any conflict perspective is more associated with human aspects than the non-human factors.

Our study followed a mixed-method approach using survey and semi-open expert interviews. The survey analysis is based on a reasonable but limited number of respondents. Besides, more than 75% of the respondents are from west Europe and north America. Hence, we acknowledge that the findings of the survey might have been biased by the region and the short number of respondents. Furthermore, the survey reflects the perception of the respondents based on the questions. In addition, the questions used for measuring conflict and software project success may be not precise and completely reflecting what they actually were measuring. Nevertheless, such limitations exist inherently in any survey study. We recommend future research to replicate and extend the survey of this study in order to broaden the knowledge about nature of conflicts in software development and increase the generalizability of its results.

To cover the limitations of our survey study, we also conducted 13 in-depth semi-structured interviews. We aimed to tackle the weaknesses of the survey and dive deeper into the rationale behind the findings of the survey. However, findings from the expert interviews are inherently limited to personal experiences of the interviewees and we cannot make firm statements about their generalizability.

We believe by combining both a survey study and an expert interview study, we reasonably tackled the limitations of both research methods and provide interesting findings for both researchers and practitioners, particularly project or product managers.

6. Conclusion and future work

In this study, we introduced a new perspective on conflicts based on STS theory to incorporate human and non-human factors. Based on this perspective, we investigated the relationship between HRC and NHRC and software project success. Our findings showed that both HRC and NHRC are negatively associated with software project success. Moreover, our investigation into the moderating effect of team size and the size of an organization led to interesting findings. Regarding team size, there is no statistically significant correlation between NHRC and software project success in large teams, whereas we found NHRC negatively correlated with software project success in smaller teams. Regarding the size of an organization, NHRC are negatively correlated with software project success in corporations, whereas there is no evidence for such a correlation in SMEs. We also found that HRC is negatively correlated with software project success, regardless of team size and the size of an organization. We conclude that even though resolving conflicts properly can have potential positive effects, presence of conflicts, particularly HRC, is an indicator for poor progress in software projects.

Future research can dive deeper into other factors that may moderate the impact of conflicts. research should also incorporate other aspects into the investigation, such as the relationship between different conflict resolution methods and the consequences of HRC and NHRC.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mohammad R. Basirati: Conceptualization, Methodology, Investigation, Writing - original draft, Writing - review & editing. **Marko Otasevic:** Investigation, Writing - original draft. **Koushyar Rajavi:** Methodology. **Markus Böhm:** Methodology, Conceptualization, Supervision, Project administration. **Helmut Krcmar:** Methodology, Supervision, Funding acquisition.

Acknowledgement

This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) collaborative research center 'Sonderforschungsbereich SFB 768 "Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products".

We also thank Morvarid Khadem and Bijan Eghtesadi for useful discussions and comments on the manuscript.

References

[1] N. Agarwal, U. Rathod, Defining 'success' for software projects: an exploratory revelation, Int. J. Project Manag. 24 (4) (2006) 358–370.
[2] A. Aldahmash, A.M. Gravell, Y. Howard, A review on the critical success factors of agile software development, in: Proceedings of the European Conference On Software Process Improvement, Springer, 2017, pp. 504–512.
[3] M. Aldekhail, A. Chikh, D. Ziani, Software requirements conflict identification: review and recommendations, Int. J. Adv. Comput. Sci. Appl. 7 (10) (2016) 326–335.
[4] A.C. Amason, H.J. Sapienza, The effects of top management team size and interaction norms on cognitive and affective conflict, J. Manag. 23 (4) (1997) 495–516.
[5] R. Atkinson, Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria, Int. J. Project Manag. 17 (6) (1999) 337–342.
[6] R.P. Bagozzi, Y. Yi, Specification, evaluation, and interpretation of structural equation models, J. Acad. Mark. Sci. 40 (1) (2012) 8–34.
[7] M. Bano, D. Zowghi, A systematic review on the relationship between user involvement and system success, Inf. Softw. Technol. 58 (2015) 148–169.
[8] H. Barki, J. Hartwick, Interpersonal conflict and its management in information system development, Mis Q. (2001) 195–228.
[9] W. Belassi, O.I. Tukel, A new framework for determining critical success/failure factors in projects, Int. J. Project Manag. 14 (3) (1996) 141–151.
[10] R.P. Bostrom, J.S. Heinen, Mis problems and failures: a socio-technical perspective, part II: the application of socio-technical theory, MIS Q. 1 (4) (1977) 11–28.
[11] Bryant, F.B., and Yarnold, P.R. 1995. "Principal-Components Analysis and Exploratory and Confirmatory Factor Analysis").
[12] R.B. Cattell, The Scree Test for the Number of Factors, Multivar. Behav. Res. 1 (2) (1966) 245–276.
[13] H.-.G. Chen, J.J. Jiang, J.-.C. Chen, J. Shim, The impacts of conflicts on requirements uncertainty and project performance, J. Int. Technol. Inf. Manag. 13 (3) (2004) 2.
[14] J. Daspit, C. Justice Tillman, N.G. Boyd, V. Mckee, Cross-functional team effectiveness: an examination of internal team environment, shared leadership, and cohesion influences, Team Perform. Manag. Int. J. 19 (1/2) (2013) 34–56.
[15] S.M. Easterbrook, E.E. Beck, J.S. Goodlet, L. Plowman, M. Sharples, C.C. Wood, A survey of empirical studies of conflict, in: CSCW: Cooperation or Conflict?, Springer, 1993, pp. 1–68.
[16] A. Fisk, N. Berente, K. Lyytinen, Boundary spanning competencies and information system development project success, ICIS (2010) 96.
[17] Y. Fried, H.A. Ben-David, R.B. Tiegs, N. Avital, U. Yeverechyahu, The interactive effect of role conflict and role ambiguity on job performance, J. Occup. Organ. Psychol. 71 (1) (1998) 19–27.
[18] B. George, T. Erikson, A. Parhankangas, Preventing dysfunctional conflict: examining the relationship between different types of managerial conflict in venture capital-backed firms, Venture Cap. 18 (4) (2016) 279–296.
[19] E. Giebels, O. Janssen, Conflict stress and reduced well-being at work: the buffering effect of third-party help, Eur. J. Work Organ. Psychol. 14 (2) (2005) 137–155.
[20] J. Gläser, G. Laudel, Experteninterviews Und Qualitative Inhaltsanalyse, Springer-Verlag, 2010.
[21] L.L. Greer, K.A. Jehn, Chapter 2 the pivotal role of negative affect in understanding the effects of process conflict on group performance, in: Affect and Groups, Emerald Group Publishing Limited, 2007, pp. 21–43.
[22] W. Guang-dong, The relationship between project team dynamic feature, conflict dimension and project success–an empirical research from Shanghai, China, Pakistan J. Stat. 29 (6) (2013).
[23] I. Hadar, A. Zamansky, Cognitive factors in inconsistency management, in: Proceedings of the IEEE 23rd International Requirements Engineering Conference (RE), IEEE, 2015, pp. 226–229.
[24] He, J. 2007. "The Moderating Effect of Cognitive Capability on Task Conflict: A Longitudinal Study of Task Conflict and Team Performance in Student Software Development Teams," *Association for Information Systems - 13th Americas Conference on Information Systems, AMCIS 2007: Reaching New Heights*, pp. 2287–2307.
[25] K.B. Hjerto, B. Kuvaas, Burning hearts in conflict: new perspectives on the intragroup conflict and team effectiveness relationship, Int. J. Confl. Manag. 28 (1) (2017) 50–73.
[26] L.t. Hu, P.M. Bentler, Cutoff criteria for fit indexes in covariance structure analysis: conventional criteria versus new alternatives, Struct. Equ.Model. Multidiscip. J. 6 (1) (1999) 1–55.
[27] L.A. Ika, Project Success as a Topic in Project Management Journals, Project Manag. J. 40 (4) (2009) 6–19.
[28] K.A. Jehn, A multimethod examination of the benefits and detriments of intragroup conflict, Adm. Sci. Q. (1995) 256–282.
[29] K.A. Jehn, A qualitative analysis of conflict types and dimensions in organizational groups, Adm. Sci. Q. 42 (3) (1997) 530–557.
[30] K.A. Jehn, E.A. Mannix, The dynamic nature of conflict: a longitudinal study of intragroup conflict and group performance, Acad. Manag. J. 44 (2) (2001) 238–251.
[31] M. Jørgensen, A survey on the characteristics of projects with success in delivering client benefits, Inf. Softw. Technol. 78 (2016) 83–94.
[32] R. Joslin, R. Müller, The impact of project methodologies on project success in different project environments, Int. J. Manag. Proj. Bus. 9 (2) (2016) 364–388.
[33] B. Kaplan, K.D. Harris-Salamone, Health it success and failure: recommendations from literature and an Amia workshop, J. Am. Med. Inform. Assoc. 16 (3) (2009) 291–299.
[34] Kawamura, T., and Takano, K. 2014. "Factors affecting the project performance of information systems development: comparison of organizational cultures," *2014 21st Asia-Pacific Software Engineering Conference*: IEEE, pp. 327–334.
[35] M. Kersten, A Cambrian explosion of devops tools, IEEE Softw. 35 (2) (2018) 14–17.
[36] S.U. Khan, M. Niazi, R. Ahmad, Empirical investigation of success factors for offshore software development outsourcing vendors, IET Softw. 6 (1) (2012) 1–15.
[37] R.B. Kline, Principles and Practice of Structural Equation Modeling, Guilford publications, 2015.
[38] L. Leite, C. Rocha, F. Kon, D. Milojicic, P. Meirelles, A survey of devops concepts and challenges, ACM Comput. Surv. 52 (6) (2019) Article 127.
[39] N. Levina, Collaborating on multiparty information systems development projects: a collective reflection-in-action view, Inf. Syst. Res. 16 (2) (2005) 109–130.
[40] T.-.P. Liang, J. Jiang, G.S. Klein, J.Y.-C. Liu, Software quality as influenced by informational diversity, task conflict, and learning in project teams, IEEE Trans. Eng. Manag. 57 (3) (2010) 477–487.
[41] T.-.P. Liang, C.-.C. Liu, T.-.M. Lin, B. Lin, Effect of team diversity on software project performance, Ind. Manag. Data Syst. 107 (5) (2007) 636–653.
[42] J.Y.-C. Liu, H.-.G. Chen, C.C. Chen, T.S. Sheu, Relationships among Interpersonal Conflict, Requirements Uncertainty, and Software Project Performance, Int. J. Proj. Manag. 29 (5) (2011) 547–556.
[43] K. Lovelace, D.L. Shapiro, L.R. Weingart, Maximizing cross-functional new product teams' innovativeness and constraint adherence: a conflict communications perspective, Acad. Manag. J. 44 (4) (2001) 779–793.
[44] Mäntylä, M.V., Jørgensen, M., Ralph, P., and Erdogmus, H. 2017. "Guest Editorial for Special Section On Success and Failure in Software Engineering." Springer.
[45] A.T. Mohr, J.F. Puck, Role Conflict, General Manager job satisfaction and stress and the performance of IJVS, Eur. Manag. J. 25 (1) (2007) 25–35.
[46] A.C. Mooney, P.J. Holahan, A.C. Amason, Don't take it personally: exploring cognitive conflict as a mediator of affective conflict, J. Manag. Stud. 44 (5) (2007) 733–758.
[47] G.C. Moore, I. Benbasat, Development of an instrument to measure the perceptions of adopting an information technology innovation, Inf. Syst. Res. 2 (3) (1991) 192–222.
[48] B. Nuseibeh, S. Easterbrook, A. Russo, Leveraging inconsistency in software development, Computer 33 (4) (2000) 24–29.
[49] N. Parolia, J.V. Chen, J.J. Jiang, G. Klein, Conflict resolution effectiveness on the implementation efficiency and achievement of business objectives in it programs: a study of it vendors, Inf. Softw. Technol. 66 (2015) 30–39.
[50] L.H. Pelled, K.M. Eisenhardt, K.R. Xin, Exploring the black box: an analysis of work group diversity, conflict and performance, Adm. Sci. Q. 44 (1) (1999) 1–28.
[51] J. Pernstål, T. Gorschek, R. Feldt, D. Florén, Requirements communication and balancing in large-scale software-intensive product development, Inf. Softw. Technol. 67 (2015) 44–64.
[52] P. Poon, C. Wagner, Critical success factors revisited: success and failure cases of information systems for senior executives, Decis. Support Syst. 30 (4) (2001) 393–418.
[53] M. Prilepok, Managing Conflict Effectively in Negotiations, Operations Extranet, 2018.
[54] D. Ribes, S. Jackson, S. Geiger, M. Burton, T. Finholt, Artifacts that organize: delegation in the distributed organization, Inf. Organ. 23 (1) (2013) 1–14.
[55] Rizzo, J.R., House, R.J., and Lirtzman, S.I. 1970. "Role Conflict and Ambiguity in Complex Organizations," *Administrative science quarterly*), pp. 150–163.
[56] J.J.L. Schepers, E.J. Nijssen, G.A.H. van der Heijden, Innovation in the frontline: exploring the relationship between role conflict, ideas for improvement, and employee service performance, Int. J. Res. Mark. 33 (4) (2016) 797–817.
[57] S. Schmidt, U. Roesler, T. Kusserow, R. Rau, Uncertainty in the workplace: examining role ambiguity and role conflict, and their link to depression—a meta-analysis, Eur. J. Work Organ. Psychol. 23 (1) (2014) 91–106.
[58] F.P. Seth, E. Mustonen-Ollila, O. Taipale, K. Smolander, Software quality construction in 11 companies: an empirical study using the grounded theory, Softw. Qual. J. 23 (4) (2015) 627–660.
[59] M. Shameem, B. Chandra, C. Kumar, A.A. Khan, Understanding the relationships between requirements uncertainty and nature of conflicts: a study of software development team effectiveness, Arab. J. Sci. Eng. 43 (12) (2018) 8223–8238.
[60] J.D. Shaw, J. Zhu, M.K. Duffy, K.L. Scott, H.-.A. Shih, E. Susanto, A Contingency model of conflict and team effectiveness, J. Appl. Psychol. 96 (2) (2011) 391.
[61] W. Song, Requirement management for product-service systems: status review and future trends, Comput. Ind. 85 (2017) 11–22.
[62] G.L. Stewart, A meta-analytic review of relationships between team design features and team performance, J. Manag. 32 (1) (2006) 29–55.
[63] B.N. Sullivan, Competition and beyond: problems and attention allocation in the organizational rulemaking process, Organ. Sci. 21 (2) (2010) 432–450.

[64] K.-.H. Tsai, T.T. Hsu, Cross-functional collaboration, competitive intensity, knowledge integration mechanisms, and new product performance: a mediated moderation model, Ind. Mark. Manag. 43 (2) (2014) 293–303.
[65] V. Venkatesh, S.A. Brown, H. Bala, Bridging the qualitative-quantitative divide: guidelines for conducting mixed methods research in information systems, MIS Q. (2013) 21–54.
[66] J.A. Wall Jr, R.R. Callister, Conflict and its management, J. Manag. 21 (3) (1995) 515–558.
[67] S. Wiesner, E. Marilungo, K.-.D. Thoben, Cyber-physical product-service systems: challenges for requirements engineering (mini special issue on smart manufacturing), Int. J. Autom. Technol. 11 (1) (2017) 17–28.
[68] L.-.R. Yang, J.-.H. Chen, X.-.L. Wang, Assessing the effect of requirement definition and management on performance outcomes: role of interpersonal conflict, product advantage and project type, Int. J. Proj. Manag. 33 (1) (2015) 67–80.
[69] M. Zahedi, M. Shahin, M. Ali Babar, A systematic review of knowledge sharing challenges and practices in global software development, Int. J. Inf. Manage. 36 (6, Part A) (2016) 995–1019.
[70] D. Zowghi, V. Gervasi, Erratum to “on the interplay between consistency, completeness, and correctness in requirements evolution”, Inf. Softw. Technol. 46 (11) (2004) 763–779.

Main Publication: P4

INTERNATIONAL DESIGN CONFERENCE - DESIGN 2018
https://doi.org/10.21278/idc.2018.0287

TOWARDS SYSTEMATIC INCONSISTENCY IDENTIFICATION FOR PRODUCT SERVICE SYSTEMS

M. R. Basirati, M. Zou, H. Bauer, N. Kattner, G. Reinhart, U. Lindemann, M. Böhm, H. Krcmar and B. Vogel-Heuser

Abstract

Value shift towards services led to emergence of product-service systems (PSS) as intertwined products and services. PSS development requires collaborating teams with higher domain diversity to tackle service side as well as product side. Since every domain employs a particular set of tools and models, it is challenging to manage consistency among them. However, the PSS literature lacks approaches for managing inconsistency among various type of models. This study proposes a framework that supports establishing a systematic solution for inconsistency identification during PSS development.

Keywords: product-service systems (PSS), model based engineering, modelling, systematic approach

1. Introduction

Competitive global economy necessitates manufacturers to increase sustainability and having a longer relationship with customers by providing new services in addition to the conventional products (Tukker, 2004). Besides, environmental considerations are putting restrictions on the products and the way they are developed leading manufacturers to rely more on value share of services (Mont, 2002; Meier et al., 2010). This shift towards services introduced concept of product-service-system (PSS) as a system consists of combined product(s) and service(s) (Baines et al., 2007).
To develop a PSS, diverse teams from managerial to technical level need to continually collaborate in order to address high level vision of the system as well as detailed functionalities and interactions between services and physical elements (Vasantha et al., 2012). Thus, not only the number of stakeholders raises in PSS development, but also the stakeholders are more likely to be heterogeneous from dissimilar domains. Accordingly, PSS development deals with multidisciplinary approaches, artefacts and models expanded over different levels of organization (Vasantha et al., 2012).
Thus, it is vital to manage consistency among the models from heterogeneous domains during a PSS development. While numerous studies addressed PSS modelling and proposed various modelling techniques such as Data Flow Diagrams (Durugbo et al., 2011) and Systems Modelling Language (Balmelli, 2007), there is lack of research on how to keep the complex network of models during PSS development consistent. Nevertheless, a consistent design process is crucial in both ensuring the system functionality and increasing interdisciplinary co-design. For example, in a digitalized machinery company, offered data services should meet requirements, allowed by the adopted sensor types and not violate country-specific privacy laws. Though various methods have been proposed to partially automate the inconsistency identification processes, a general and systematic underlying framework is missing (Zou and Vogel-Heuser, 2017).

Therefore, in this study, we investigate on inconsistencies in PSS development and develop a framework, which can be applied to identify the inconsistencies systematically. The reminder of this paper is structured as follows: Section 2 provides related work on inconsistency management as well as PSS modelling literature. Afterwards, we introduce different types of inconsistencies in Section 3. Subsequently the developed framework to identify inconsistencies is presented in Section 4. The framework is applied and evaluated on a case of e-bike-sharing system, presented in Section 5. We discuss the framework applicability and its limits in Section 6. Finally, Section 7 summarize the study and shed light on future works regarding managing inconsistencies during PSS development.

2. Related work

Inconsistency Management by definition is the process of handling dependencies in a way that the goals of stakeholders are concerned (Spanoudakis and Zisman, 2001). We reviewed the literature on inconsistency management from software engineering domain as well as mechanical engineering studies. Besides, we present few number of studies, which addressed inconsistency-related topics in PSS development.

Nuseibeh et al. (2000) and Finkelstein et al. (1996) have developed processes for the management of inconsistencies in the domain of software engineering. Both approaches share the common feature that inconsistencies must be determined with respect to defined consistency rules. The consistency rules, which are defined by experts, specify two essential necessities. First, they check whether a model is applicable as a valid model of a particular modelling language or not. In another words, this step analyses correctness of the models from syntactical point of view. Second, they ensure that the software models in a development process comply with valid standards of the software development project. In addition, both approaches involve activities to detect, diagnose and manage inconsistencies. There are other supporting activities, such as the specification and application of an inconsistency policy. These additional activities are not included in both approaches, which makes them different from each other. Besides, Finkelstein focuses on the term "inference", expressed as conflicts of interdependencies.

Spanoudakis and Zisman (2001) combine the both aforementioned approaches (Finkelstein et al., 1996; Nuseibeh et al., 2000) to create a process concentrated on special methods and techniques for inconsistency management. The approach is represented in six steps, which starts with the detection of overlaps, followed by the detection and the diagnosis of inconsistencies, handling of inconsistencies and finally, the specification and application of an inconsistency management policy.

Gausemeier et al. (2009) employ triple graph grammars and introduce a rule-based approach aimed at preserving consistency between domain-spanning and domain-specific models in the development of complex mechatronic systems. For this purpose, they developed an approach based on triple graph grammars to capture correspondences between models. However, this approach is limited to a special technique and its applicability on cross-domain models is weak.

Hehenberger et al. (2010) define consistency rules and propose a conceptual approach based on the automatic checking of the consistency rules. First, they use domain-spanning ontologies for identifying overlaps. Subsequently, they apply the consistency rules, which are simple conditions of a model that are either true or false, depending on whether the model satisfies them or not.

Qamar et al. (2012) address dependency modelling and present an approach to avoid inconsistencies in models, based on the explicit modelling of dependencies. They argue that impact of changing a model on the other models can be predicted by modelling dependencies, which enables easier inconsistency management.

Herzig and Paredis (2014) apply pattern matching, which is based on the transformation of models into graphs and a subsequent pattern-based identification of inconsistencies.

Feldmann et al. (2015) apply semantic web technology and present an approach where properties of semantic webs, which enable processing not only the linking of data but also their meaning, can be used to identify inconsistencies in models. A knowledge-base approach represents all Models in a Resource Description Framework (RDF) and reveals inconsistencies with SPARQL inconsistency query.

Dávid et al. (2016) introduce the inconsistency tolerance. In the context of this approach, semantic inconsistencies are quantified. Afterwards, it is decided how far they can be tolerated.

In PSS field, Shimomura and Hara (2010) introduce a conflict resolution method for PSS development. The method consists of two major strategies. First, an inconsistency is detected by lexical analysis of names and descriptions of functions and variables. If two objects' names include related words, the method investigates if there is an inconsistency or not. The second strategy applies a set-based approach to detect value overlaps that lead to inconsistencies. To this end, the related objects are identified manually. Nevertheless, the study does not address how heterogeneity of PSS models are handled.
Song and Sakao (2016) investigate on conflicts among services in product-service offerings, which also employs a linguistic techniques. However, the study does not cover the cross-models inconsistency problem.

3. Inconsistency types

In this study, we define inconsistency as any logical contradiction between two facts or two presentations of facts expressed in formal models as well as in informal artefacts such as requirements written in natural language. This definition excludes any high level conflict between goals, priorities and plans of people or organizational units. Therefore, we focus specifically on how heterogeneous models and artefacts can lead to inconsistencies during PSS development.
To this end, we developed a classification of inconsistencies that specifies what types of inconsistencies exist between models by extending the work of Feldman et al. (2015). The inconsistencies are classified based on two high level dimensions. First dimension determines how the two inconsistent elements are related to each other. We defined four high level relationship types, namely existence, equivalency, refinement and satisfaction. Furthermore, we determine if an inconsistency is occurred on syntactical level or semantical level. The syntactical level consists of notational and conventional types. The semantical level is divided to system/project specific as well as domain-specific types. The inconsistency types and examples are presented in Table 1. Besides, concrete examples of inconsistency types are demonstrated based on a case in the discussion, Section 6.

Table 1. Inconsistency types

	Existence	Equivalence	Refinement	Satisfaction
Notational	Missing Name; Typo	Different Names for Same Element	-	-
Conventional	Missing a Standard's Element	Incompliancy with a Standard's Element	-	-
System/Project-specific	Missing Element in a Model based on System's Logic	Different Value for Same Element in Different Models	Different Refined Value for an Element	Unsatisfied Logics of System
Domain-specific	Missing Element in a Model based on General Domain Rules	Different Value for an Element based on Domain Rules	Wrong Refinement based on Domain Rules	Unsatisfied Domain Rules

If an element is missing, which is supposed to exist in the model, it is considered as an existence inconsistency. For example, missing a name, a structural section in document or un-modelled part of a system can be assigned to this group. Similarly, if two elements are supposed to be equal, but they are not, there is an equivalence inconsistency. However, more complicated types of inconsistencies can be identified. If an element is refined version of another, but the refinement is done incorrectly, it can be recognized as a refinement inconsistency. For example, a Simulink model can be refined version of MATLAB codes. Finally, an inconsistency can arise, when an element is supposed to satisfy another element's conditions, domain-specific rules or project-specific rules.
Based on the other dimension, inconsistency can violate syntactical or semantical rationale. Notational instances are a typo or inconsistent use of terms for a same element. Besides, there might exist a set of conventional rules regarding the syntactic of a model or artefact. Various conventional rules can be set up including standards, guidelines, and quality attributes of specifications and so on. Accordingly, inconsistencies can violate conventional rules such as syntax of a modelling language or incompliancy

with a standard. System/project-specific inconsistencies are the ones that contradict logic and function of the system or project, which are under development. For example the relationship between components in the system, the specific defined values and so on. On the other hand, domain-specific inconsistencies are discipline-specific and root in general knowledge related to a domain, an example can be violating a physical law.

4. Framework for inconsistency identification in PSS

In this section, we describe a holistic framework that tackles the overall procedure for inconsistency identification in PSS development. Besides, there are many parameters that determines how an inconsistency identification procedure will be realised. For example it should be determined to what extent the process is performed automatically. Therefore, the framework includes such parameters and we explain them subsequently in the following.

4.1. Process of inconsistency identification in PSS development

To identify inconsistencies during PSS development, the framework considers five high level phases that need to be carried out. Since many departments and teams are involved in PSS development, it is necessary to first trace the information flow among them. Based on the information flow, we would be able to identify related models and artefacts as well as how they are related to each other. Subsequently, the dependencies among related models are detected in the next phase. At this stage, we have the related models and the dependencies among them. However, as the models are from heterogeneous domains and teams, there are expressed in different languages and forms. Therefore, a mutual form is needed to consolidate the information and enable comparing values. This is done in the fourth phase and finally, in the last phase, dependent information in the same form can be analysed with regard to some rationale and the inconsistencies will be exposed.

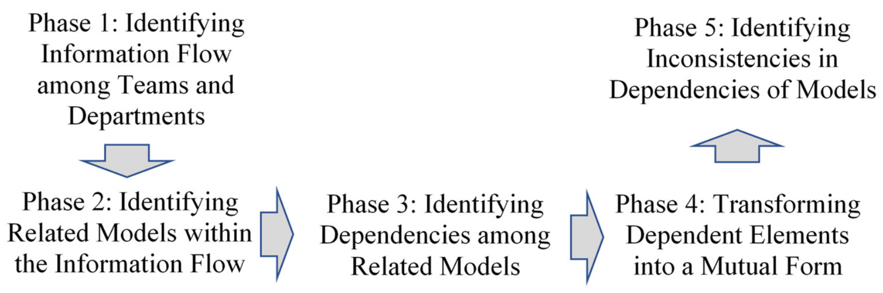

Phase 1: Identifying Information Flow among Teams and Departments
Phase 2: Identifying Related Models within the Information Flow
Phase 3: Identifying Dependencies among Related Models
Phase 4: Transforming Dependent Elements into a Mutual Form
Phase 5: Identifying Inconsistencies in Dependencies of Models

Figure 1. Inconsistency identification process

The described major phases for PSS inconsistency identification can be implemented by various methods and tools. For example, in the first and second phase, communication data of stakeholders can be investigated using information retrieval techniques or manually organizational structure of the project can be analysed. Reviewing such methods and tools for every phase is out of scope of this study and we address it in future work. However, the procedure for identifying inconsistencies among related models (phases 3 to 5) can be detailed more without lacking its general applicability.
To this end, we describe what steps are needed to be taken in every phase. These steps are elaborated based on three abstraction levels, namely meta-model, model and reality. As depicted in Figure 2, after identifying a dependency on model level, the exact elements which create the dependency have to be specified on model level. Afterwards, in order to systematically transform information from dissimilar models into a mutual form, it is required to trace dependency on the meta-model level. Specifying details on a meta-model level enables practitioners to generalize dependencies as well as inconsistencies and repeat the process systematically. In particular, such abstraction levels are essential for automating the procedure using defined algorithms for machine. After the relationship between two elements are identified on the meta-model level, the dependent meta-model level elements should be mapped to a corresponding element on the meta-model level of a mutual form. Consequently, we would be able to

capture the information in the same format. In the final phase, we need to reason inconsistencies based on some rationale. However, logics are expressed in a general manner such as domain-specific laws or system logics. Therefore, the rationale can be formulated on the meta-model level and ground the final identification as the final step.

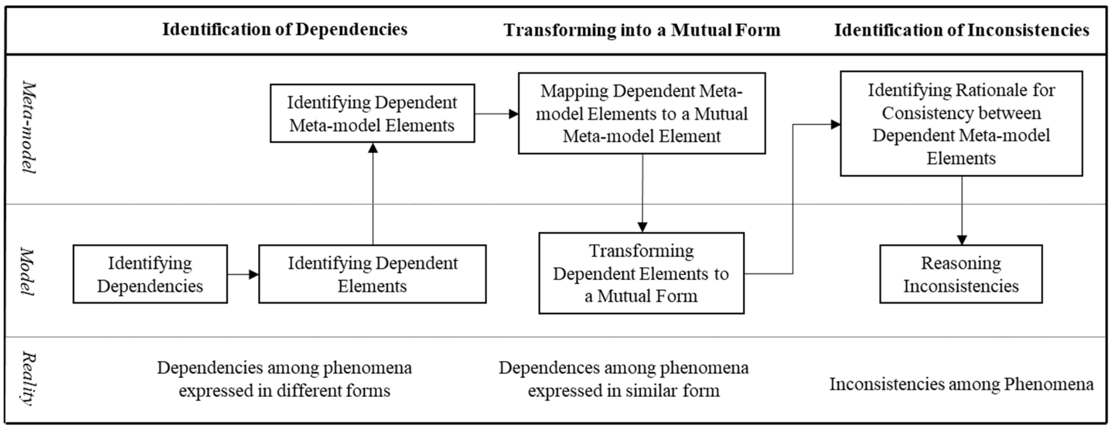

Identification of Dependencies
Transforming into a Mutual Form
Identification of Inconsistencies
Meta-model
Model
Reality
Identifying Dependent Meta-model Elements
Mapping Dependent Meta-model Elements to a Mutual Meta-model Element
Identifying Rationale for Consistency between Dependent Meta-model Elements
Identifying Dependencies
Identifying Dependent Elements
Transforming Dependent Elements to a Mutual Form
Reasoning Inconsistencies
Dependencies among phenomena expressed in different forms
Dependences among phenomena expressed in similar form
Inconsistencies among Phenomena

Figure 2. Inconsistency identification through abstraction levels

4.2. Parameters of inconsistency identification in PSS development

In addition to the process itself, there are parameters that determine how the process is realised. These parameters are related to the model and targeted inconsistency as well as the details of identification method. We distinguish between problem-side and solution-side parameters and explain them in detail. The problems-side parameters characterize the inconsistency and the situation, in which the consistency may emerge, while the solution-side parameters represent the techniques and technologies that are used to identify the inconsistency.

The framework addresses degree of formality, degree of criticalness and inconsistency type as problem-side parameters. Degree of formality determines to what extent the involved models in an inconsistency are formal. For example, mathematical equations have high formality, while artefacts written in natural language, such as user requirements can be considered as the most informal models. The degree of formality influences on how dependent elements can be identified and transformed into a mutual form. For example, graph-based formal models such as UML are easier to be mapped into a meta-model and accordingly transformed into a mutual form. However, informal models require different mechanisms. In particular, automation of the process in case of formal models is more straightforward than informal ones. Degree of Criticalness determines to what extent the targeted inconsistency is hazardous to the system and the project. A high critical inconsistency should not be missed by the implemented identification methods. Therefore, proper approaches and tools need to be employed in order to reach high recall for such inconsistences. Furthermore, dissimilar approaches can be employed based on the inconsistency type as it specifies the relationship between two elements and the rationale behind an inconsistency.

Similarly, there are three solution-side parameters that address general features of a solution for inconsistency identification in PSS development. First, considering the parameters of problem-side, degree of automation, based on which identification process is developed should be decided. How much automation is required directly impacts on the technologies and tools that are applied. In addition, there are in general two types of automatically re-employing acquired knowledge, formulating the knowledge into a set of rules or applying machine learning algorithms to seizure knowledge into a re-applicable model. These two types of automatic solutions necessitate different mechanisms and infrastructures. Thus, applicability of both methods should be analysed based on the PSS development settings and requirements. Finally, appropriateness of different technologies for the mutual form, to which the information from heterogeneous models are transformed, should be evaluated. Although this parameter determines how the overall process is realised, the other parameters such as degree of formality and degree of automation influence on deciding it. For example, the technology of mutual form needs to be suited to the involved models as well as the tools and technologies used in automatizing the process.

Table 2. Influencing parameters for inconsistency identification

Problem-side	Solution-side
Degree of Formality	Degree of Automation
Degree of Criticalness	Rule-based vs. Machine Learning
Type of Inconsistency	Mutual Form Technology

5. Application: An inconsistency case

We describe application of the framework based on an e-bike-sharing system as an example of PSS. The e-bike-sharing system, called PSSycle, is developed by a group of students in context of an interdisciplinary research project that addresses cyclic innovation in PSS development. The large structure of the research project allows us to apply entire process of the framework and consider the complete relevant aspects. In the following, we describe the case and an instance of inconsistency, afterwards we apply the framework on identifying such an inconsistency.

5.1. PSSycle: An E-Bike-Sharing PSS

The research project consists of 7 research departments including Information Systems, Product Development, Mechatronic Design, Manufacturing Planning, Organizational Psychology, Automatic Control and Technology in Society. PSSycle development included all departments except Organizational Psychology, as they were focused on unrelated issues to this case. We briefly introduce the responsibility of every department in the development process. Department for Technology in Society investigates the needs for a PSS in society. Information System department tackled software parts of PSSycle as an e-bike-sharing system, including board computer and mobile app. Departments for Mechatronic Design and Product Development designed and developed the frame, the lock and the final assembly. The department for Manufacturing Planning modelled how the PSSycle will be manufactured in a real industrial settings. Besides, Automatic Control department was responsible for simulation of PSSylce.

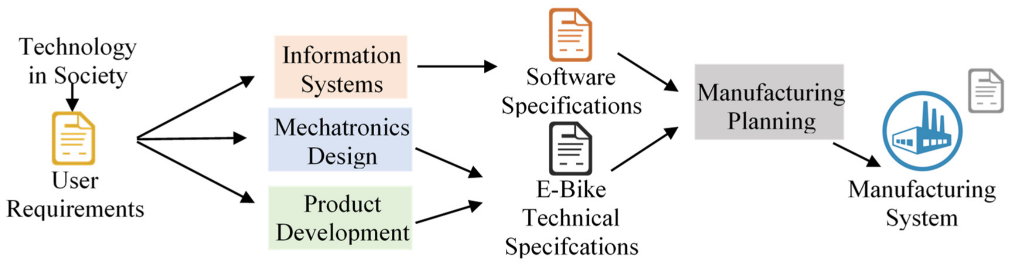


Figure 3. Information flow for PSSycle development

In an exemplary scenario, Technology in Society department finds out that there are two types of PSSycle users. A group of users ride e-bikes for very short trips, while another group, ride e-bikes for longer trips. However, the current e-bikes cannot ride such long trips completely, supported by the battery power. Therefore, from a business perspective it is decided that PSSycle should establish new services for customizing e-bikes for different purposes. As the first step, two types of e-bikes are offered based on their range: short-range (regular ones) and long-range. To this end, Departments for Mechatronics Design and Product Development decide to add a new e-bike with the same specifications as regular ones, but with a higher capacity battery, which has a bigger size than the previously used battery type for short-range e-bikes. For manufacturing such an e-bike, the gripper used in manufacturing line must support the size of the batteries. We apply the framework on the described situation and elaborate how an inconsistency can be identified.
As first step of the framework, we identify the information flow among the teams and departments in case of the described situation. Depicted in Figure 3, user requirements are formulated by Technology

in Society department. Afterwards, the user requirements are analysed, designed and translated into specifications. The e-bike specifications are detailed by departments for Mechatronics Design and Product Development. Besides, department for Information Systems delivers software specifications. Subsequently, department for Manufacturing Planning models how the e-bikes should be manufactured in an industrial settings.

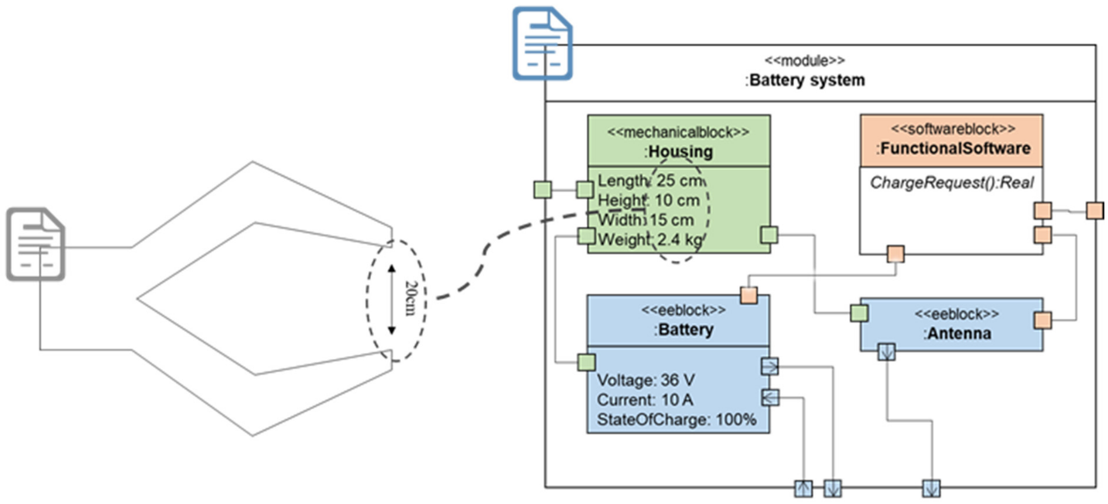


Figure 4. Example of model dependency for PSSycle

Based on the second step of the framework, we identify every department's models related to the battery manufacturing (we present several models as examples, however a complete model set is out of scope of this study). User requirements expressed in natural language can be considered as a model developed by Technology in Society department. UML models are from Information Systems Department. SysML4Mechatronics (Kernschmidt and Vogel-Heuser, 2013) and CAD models are respectively from departments for Mechatronics Design and Product Development. Finally, digital factory models are developed by department for Manufacturing Planning.
In the next step, dependencies among the models need to be identified. Based on the aforementioned scenario, we demonstrate a dependency between SysML4Mechatronics model of the e-bike battery and the gripper's model (depicted in Figure 4). Dependent elements are the gripper's supporting width and the battery's width and length, which it is planned to move.

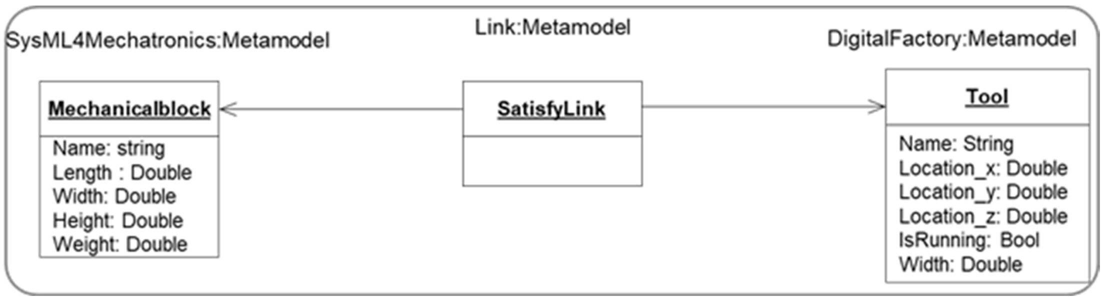


Figure 5. Meta-model level dependency

After identifying dependencies, the dependent elements should be transformed into a mutual form. This is an essential step for an automated approach. However, in a manual inspection approach, it depends on complexity of the dependent models to include or skip the phase because of high simplicity. In case of PSSycle, dimensions of the e-bike are specified in mechanicalblock of the SysML4Mechatronics model and the gripper's width is expressed in tool component of a digital factory model. Therefore, mechanicalblock and tool component are the dependent meta-model elements. We selected RDF for mutual form that dependent elements are transformed to. In the following, we show how models are formulated in RDF.

```
<rdf:Description about="http://PSSycleModels/BatterySpec">
  <mechanicalblock:name>Battery</mechanicalblock:name>
  <mechanicalblock:length>25</mechanicalblock:length>
  <mechanicalblock:width>10</mechanicalblock:width>
  <mechanicalblock:height>15</mechanicalblock:height>
  <mechanicalblock:weight>2.4</mechanicalblock:weight>
</rdf:Description>

<rdf:Description about="http://PSSycleModels/GripperModel">
  <digitalfactorytool:name>Gripper</digitalfactorytool:name>
  <digitalfactorytool:location_x>1000</digitalfactorytool:location_x>
  <digitalfactorytool:location_y>1500</digitalfactorytool:location_y>
  <digitalfactorytool:location_z>1300</digitalfactorytool:location_z>
  <digitalfactorytool:width>20</digitalfactorytool:width>
</rdf:Description>
```

Figure 6. Transformed models into a mutual form (RDF)

In the final phase, consistency of dependent elements' values are investigated. To this end, we need to identify rationale that define whether a situation is in the state of consistency or inconsistency. The rationale capture general laws, logics and considerations, consequently, they can be assigned to meta-model level. In case of our example, the gripper must be able to pick up and move the battery from its either length or width dimension. Therefore, width of the gripper have to be equal or greater than the battery's length and width. This can be expressed on the meta-model level more formally as:

(mechanicalblock:width≤digitalfactorytool:width)AND(mechanicalblock:length≤digitalfactorytool:width)

Based on such a rule on the meta-model level, an inconsistency can be identified between length of the battery and the gripper's width. The inconsistency on the model level is demonstrated in Figure 7.

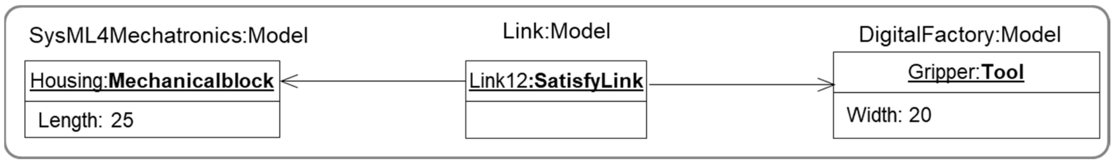


Figure 7. PSSycle inconsistency example on model level

6. Discussion

In the previous section, we projected how overall process of the framework can support practitioners to systematically identify inconsistencies between heterogeneous models of PSS. Besides, we briefly showed an RDF example of transforming dissimilar models into a mutual form. In this section, based on the PSSycle case, we explain introduced parameters of the framework. Moreover, we discuss the limitations of the framework.

The battery was modelled in SysML4Mechatronics, which offers a high degree of formality. Hence, it allows us to adopt higher degree of automation. The inconsistency between the battery size and the gripper belongs to satisfaction and system/project-specific types (described in Section 3). In another words, the exposed inconsistency violated a satisfaction relation defined by the system development's logics. Regarding degree of criticalness, an inconsistency's criticalness depends on real settings of a project and the related practitioner's view. Therefore, it cannot be assessed objectively. However, future work can investigate on the factors that increase criticality of an inconsistency as well as measuring the criticality based on such factors.

The solution described in the exemplary case can be implemented highly automated. Because the models are mostly formal and it is elaborated in the previous section that how the dependent elements are related in the model as well as the meta-model level. Therefore, we consider high degree of automation for the presented example. Besides, we applied a rule-based method by defining a rule that determines whether the gripper is compatible with the battery size or not. Finally, as the mutual form technology we used RDF.

Nevertheless, in the PSSycle case scenario many other types of inconsistency could emerge. For instance, a conventional inconsistency could arise if width of the gripper was expressed in inches instead of centimetres. An example of refinement inconsistencies can emerge between RDF presentation of the battery and its SysML4Mechatronics model, as the RDF is refined version of the SysML4Methcatronics model. Therefore, if the battery's width in RDF presentation is different to its value in SysML4Mechatronics, there is a system/project-specific refinement inconsistency. Besides, more complicated inconsistencies can happen between requirements collected from users and system specifications and functionalities. For example, if the battery's discharge rate is faster than the expectations of the users, this can be considered as an inconsistency.

Although, the inconsistency example of this study is rather simple, however in industrial settings with high number of interconnected elements, the need for a systematic approach to identify even trivial inconsistencies is crucial. This study establishes the fundamentals of such an approach that can be customised based on the needs and situations. For example, to increase dependability, formal methods can be employed to identify inconsistencies based on the framework. Hence, more in detail methods and analysis are required to address the introduced phases and parameters of inconsistency identification for PSS.

Furthermore, we cannot claim that the proposed framework includes all related aspects and parameters, however the high influential concepts, which are covered by the framework can be extended by future research through adding new parameters or customising the framework for a particular inconsistency problem. Moreover, the framework is developed from a PSS development perspective, nevertheless, future research can investigate and extend its applicability for general inconsistency identification purposes. Besides, the proposed framework only concentrates on the identification process and how the exposed inconsistencies should be handled is not tackled, which should be investigated in future research.

7. Conclusion

PSS development requires collaboration of diverse teams employing varied models and artefacts. Thus, there is a higher chance for occurrence of inconsistency among the models. Consequently, a systematic approach is required to support practitioners in order to identify the heterogeneous models' inconsistences during PSS development. To this end, first we introduced a classification of inconsistency types, based on which different methods and policies can be employed. Subsequently, we developed a holistic framework including the general procedure and influencing parameters of inconsistency identification. The framework can be customised based on different settings and needs. Based on an e-bike sharing system case, we applied the framework to identify an exemplary inconsistency. The case demonstrated a concrete problem of the profile fitness among different models, which is easy if it is designed by the same models and tools. However, it is complicated in the design phase of PSS, where diverse stakeholders work in parallel and distributed.

The proposed framework of this study addresses the basics of inconsistency identification. Thus, more studies required to investigate on every introduced aspect of inconsistency identification for PSS. To this end, we will apply the framework on a more complex PSS scenario to expose applicability of the framework more in detail. A future work can study what policies are more suited for different types of inconsistencies. Besides, most of the previous studies applied rule-based methods in order to detect inconsistencies among models, however, with extensive availability of data, the applicability of machine-learning techniques on models' inconsistency identification problem should be investigated. Hence, in future work, we plan to implement such a solution on higher number of models.

Acknowledgement

This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) collaborative research centre ‘Sonderforschungsbereich SFB 768 “Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products”.

References

Baines, T.S., Lightfoot, H.W., Evans, S., Neely, A., Greenough, R. et al. (2007), “State-of-the-art in product-service systems.” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* Vol. 221 No. 10, pp. 1543-1552. https://doi.org/10.1243/09544054JEM858

Balmelli, L. (2007), “An overview of the systems modeling language for products and systems development”, *Journal of Object Technology*, Vol. 6 No. 6, pp. 149-177. https://doi.org/10.5381/jot.2007.6.6.a5

Dávid, I., Syriani, E., Verbrugge, C., Buchs, D., Blouin, D. et al. (2016), "Towards inconsistency tolerance by quantification of semantic inconsistencies", *First International Workshop on Collaborative Modelling in MDE COMMitMDE 2016, Saint Malo, France, 2016*, CEUR-WS, pp. 35-44.

Durugbo, C., Tiwari, A. and Alcock, J.R. (2011), "A review of information flow diagrammatic models for product–service systems", *The International Journal of Advanced Manufacturing Technology,* Vol. 52 No. 9, pp. 1193-1208. https://doi.org/10.1007/s00170-010-2765-5

Feldmann, S., Herzig, S.J.I., Kernschmidt, K., Wolfenstetter, T., Kammerl, D. et al. (2015), "Towards effective management of inconsistencies in model-based engineering of automated production systems", *IFAC-PapersOnLine*, Vol. 48 No. 3, pp. 916-923. https://doi.org/10.1016/j.ifacol.2015.06.200

Finkelstein, A., Spanoudakis, G. and Till, D. (1996), "Managing interference", *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops, San Francisco, California, United States*, ACM, New York, NY, pp. 172-174. https://doi.org/10.1145/243327.243646

Gausemeier, J., Schäfer, W., Greenyer, J., Kahl, S., Pook, S. and Rieke, J. (2009), "Management of cross-domain model consistency during the development of advanced mechatronic systems", *Proceedings of the 17th International Conference on Engineering Design (ICED 09),* The Design Society, Glasgow.

Hehenberger, P., Egyed, A. and Zeman, K. (2010), "Consistency Checking of Mechatronic Design Models", *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - 2010, Montreal, Quebec, Canada, August 15-18, 2010*, ASME, New York, NY, pp. 1141-1148. https://doi.org/10.1115/DETC2010-28615

Herzig, S.J.I. and Paredis, C.J.J. (2014), "A Conceptual Basis for Inconsistency Management in Model-based Systems Engineering", *Procedia CIRP*, Vol. 21, pp. 52-57. https://doi.org/10.1016/j.procir.2014.03.192

Kernschmidt, K. and Vogel-Heuser B. (2013), "An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering", *2013 IEEE International Conference on Automation Science and Engineering (CASE),* IEEE, pp. 1113-1118. https://doi.org/10.1109/CoASE.2013.6654030

Meier, H., Roy, R. and Seliger, G. (2010), "Industrial product-service systems—IPS 2." *CIRP Annals-Manufacturing Technology,* Vol. 59 No. 2, pp. 607-627. https://doi.org/10.1016/j.cirp.2010.05.004

Mont, O.K. (2002). "Clarifying the concept of product–service system." *Journal of Cleaner Production,* Vol. 10 No. 3, pp. 237-245. https://doi.org/10.1016/S0959-6526(01)00039-7

Nuseibeh, B., Easterbrook, S. and Russo, A. (2000), "Leveraging inconsistency in software development", *Computer*, Vol. 33 No. 4, pp. 24-29. https://doi.org/10.1109/2.839317

Qamar, A., Paredis, C.J.J., Wikander, J. and During, C. (2012), "Dependency Modeling and Model Management in Mechatronic Design", *Journal of Computing and Information Science in Engineering*, Vol. 12 No. 4, pp. 41009. https://doi.org/10.1115/1.4007986

Shimomura, Y. and Hara, T. (2010). "Method for supporting conflict resolution for efficient PSS development", *CIRP Annals-Manufacturing Technology,* Vol. 59 No. 1, pp. 191-194. https://doi.org/10.1016/j.cirp.2010.03.122

Song, W. and T. Sakao (2016), "Service conflict identification and resolution for design of product–service offerings", *Computers & Industrial Engineering,* Vol. 98, pp. 91-101. https://doi.org/10.1016/j.cie.2016.05.019

Spanoudakis, G. and Zisman, A. (2001), "Inconsistency management in software engineering: Survey and open research issues", In: Chang S.K. (Ed.), *Handbook of software engineering and knowledge engineering*, Vol. 1: Fundamentals, pp. 329-380. https://doi.org/10.1142/9789812389718_0015

Tukker, A. (2004), "Eight types of product–service system: eight ways to sustainability? Experiences from SusProNet", *Business Strategy and the Environment*, Vol. 13 No. 4, pp. 246-260. https://doi.org/10.1002/bse.414

Vasantha, G.V.A., Roy, R., Lelah, A. and Brissaud, D. (2012), "A review of product–service systems design methodologies", *Journal of Engineering Design*, Vol. 23 No. 9, pp. 635-659. https://doi.org/10.1080/09544828.2011.639712

Zou, M. and Vogel-Heuser, B. (2017), "Feature-based Systematic Approach Development for Inconsistency Resolution in Automated Production System Design", *13th Conference on Automation Science and Engineering (CASE), Xi'an, 2017*, pp. 687-694. https://doi.org/10.1109/COASE.2017.8256183

Mohammadreza Basirati, Msc.
Technical University of Munich, Information Systems
Boltzmannstraße 3, 85748 Garching, Germany
Email: basirati@in.tum.de

Not Included in Review and Evaluation: P1

IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation

Completed Research Paper

Mohammad R. Basirati
Technical University of Munich
Boltzmannstraße 3, 85748 Garching
basirati@in.tum.de

Jörg Weking
Technical University of Munich
Boltzmannstraße 3, 85748 Garching
joerg.weking@in.tum.de

Sebastian Hermes
Technical University of Munich
Boltzmannstraße 3, 85748 Garching
sebastian.hermes@in.tum.de

Markus Böhm
Technical University of Munich
Boltzmannstraße 3, 85748 Garching
markus.boehm@in.tum.de

Helmut Krcmar
Technical University of Munich
Boltzmannstraße 3, 85748 Garching
krcmar@in.tum.de

Abstract

Nowadays, product-service systems (PSS) as an integrated system of physical products and services play a crucial role in sustainable economies. In addition to high competitive global economy, emergence of new digital paradigms is supporting the shift towards servitization. Although the great potential of such paradigms are recognized by both practice and research, their implications for PSS is not clear yet. Particularly, features of Internet-of-Things (IoT) such as total connectedness and ubiquity of smart sensors and actuators provide various new opportunities for PSS. This study explores such opportunities by conducting structured literature review and 13 interviews. We formulate the findings into two folds. First, we introduce four degrees of IoT involvement in PSS business models and we elaborate the opportunities that they create for different types of PSS. Second, we present the key technologies and approaches, which IoT provides with regard to PSS lifecycle management.

Keywords: Product-Service System, Internet-of-Things, IoT Integration, Review

Introduction

Firms have to increase their share of service offerings in order to survive in the global competitive economy (Mont 2002). Products are no more the main contributors to value creation, as the value is shifting towards services. We can see this shift in gross domestic product (GDP) of most developed countries, which are more dependent on services than physical products (Meier et al. 2010). Consequently, more service-oriented business models have emerged such as product-service systems (PSS). Most definitions of PSS describe it as a system that integrates products and services in order to create a competitive solution (Beuren et al. 2013). Furthermore, some definitions also emphasize on the role of PSS for reaching sustainability with regard to environmental and social considerations (Baines et al. 2007; Maxwell et al. 2006).

Emergence of advanced digital paradigms such as Internet of Things (IoT) is providing even more opportunities for innovative service offerings and PSS design (Kowalkowski et al. 2015; Lightfoot et al. 2013; Ulaga and Reinartz 2011). IoT is a concept for network of objects, which can sense, communicate, store data and interact with the environment (Patel and Patel 2016). IoT allows not only monitoring state of the physical objects, but it establishes the ground for progressive services such as optimization and atomization of product operations and services (Adrodegari and Saccani 2017; Porter and Heppelmann 2014).

There is a consensus among previous studies on the relevance of digital technologies such as IoT on servitization, and particularly PSS (Exner et al. 2017; Marilungo et al. 2017; Shih et al. 2016). In practice, however, adoption of IoT is a challenging issue as it requires an intensive reconfiguration of existing settings (Marilungo et al. 2017). Research, on the other hand, does not provide clear guidance on how we can exploit IoT to successfully design and develop PSS despite the need (Kiel et al. 2017). Hence, this study addresses the following research question:

Research question: What opportunities does IoT provide for PSS design and development?

We focus on two important aspects of PSS development: (1) Integrating IoT in PSS business models (2) Integrating IoT in PSS lifecycle. To get a wide-ranging understanding of IoT and PSS in research and practice, we use a structured literature review (Webster and Watson 2002a) and expert interviews (Gläser and Laudel 2010). The results provide a comprehensive overview on ideas and practices that IoT delivers for innovative PSS design and development. With regard to business development aspect, a framework elaborates the implications of different degrees of IoT involvement for different types of PSS. Furthermore, we present the core concepts and technologies, which IoT enables and can be employed to facilitate PSS lifecycle management.

Theoretical Background

PSS

PSS refers to a strategic business model design intended to integrate and combine products, services and communication based on changing costumer and stakeholder demands (Beuren et al. 2013). The concept was introduced in 1999 as a promising business model for "sustainable economic growth" (Baines et al. 2007; Maleki et al. 2017). Most articles investigating PSS rely on the definition of Goedkoop et al. (1999), who stress: *"A Product-Service System is a marketable set of products and services capable of jointly fulfilling a user's need. A Product is a tangible commodity, manufactured to be sold. A Service is an activity (work), often done on a commercial basis and for others with an economic value. A System is a combination of elements including their relations."*

Table 1 . PSS Types According to Reim et al. (2015)

	Product-oriented	Use-oriented	Result-oriented
Value Creation	Provider takes responsibility for the contracted services.	Provider is responsible for the usability of the product or service.	Provider is responsible for delivering results.
Value Delivery	Provider sells and services the product sale and service (e.g., maintenance or recycling).	Provider assures the usability of the physical product along with service.	Provider actually delivers result.
Value Capturing	Customer pays for physical product and for the performed services.	Customer can make continuous payments over time (e.g., leasing).	Customer payments are based on outcome units; that is, they pay for the result.

The PSS literature has recognized the importance of implementing integrated product-service offerings, considering them as a powerful source of competitive advantage and sustainability (Ardolino et al. 2016; Schuh et al. 2016). PSS has proven to provide advantages such as higher profit margins, new growth opportunities in saturated markets and long-term customer relationships. Besides advantages for PSS providers, PSS also benefits consumers, the environment and the society (Beuren et al. 2013). Nonetheless, PSS implementation can be challenging and lead to inconsistencies among heterogeneous teams and developing artefact (Basirati et al. 2018). Moreover, PSS adoption into existing business models is not a straightforward procedure and requires applying proper strategies (Weking et al. 2018).

Within the PSS research stream, three types of PSS have emerged. Namely, product-oriented, use oriented and result-oriented PSSs (Baines et al. 2007; Tukker 2004; Yang et al. 2009). This classification is widely accepted in the literature. Table 1 describes the three different categories of PSS with regard to their underlying business model elements (Reim et al. 2015).

Another way of looking at the three types of PSS is how far they are on the innovation scale: result-oriented PSS is the most innovative and product-oriented PSS is the least innovative. To evolve from product-oriented to result-oriented PSS, there are incremental paths and radical paths. Incremental innovation in this context means that product-oriented PSS evolves slowly to use-oriented and then further to result-oriented. This happens through a slow and steady continuous improvement process. Radical innovation, on the other hand, means that product-oriented PSS transforms directly into result-oriented PSS, skipping the use-oriented stage. This often involves a radical shift in technology and leads to a total reconfiguration of the PSS (Jing, 2012, p. 791).

IoT

The term "Internet of Things" was introduced by Kevin Ashton in a presentation in 1998 (Perera, Zaslavsky, Christen, & Georgakopoulos, 2014) and is now a technological concept with wide areas of application (Tao et al. 2014). However, there is yet no standard definition for IoT due to the fact that research about IoT is still in its infancy. Building of the seminal work of Gubbi et al. (2013), we define IoT as: *"Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large scale sensing, data analytics and information representation using cutting edge ubiquitous sensing and cloud computing." (Gubbi et al. 2013).*

The concept of Internet of things (IoT) includes both technology and services that are based on connected objects and the use of the collected data (Čolaković and Hadžialić 2018). Everyday objects can be equipped with sensors and actuators to communicate, generate and process data (Whitmore et al. 2015). Usually an object, also called a thing, communicates over network protocols with a service in the cloud (Guth et al. 2018).

We elaborate the essential components of IoT comprised within a four-layered technology stack, which comprises the insights of Bandyopadhyay and Sen (2011), Porter and Heppelmann (2014; Vuppala and Kumar (2014), Lee et al. (2013), Georgakopoulos and Jayaraman (2016), Mazhelis et al. (2012) and Wortmann and Flüchter (2015). Table 2 illustrates the multiple technology layers. The layers are independent, which means that all components can be developed independently. The communication between the components ideally proceeds through well-defined interfaces and a shared cloud-based platform. In general, the two lower layers are responsible for data capturing, where the data is generated and collected by the low-end sensor nodes. The two upper layers are contributing to data processing and data utilization in applications.

Table 2 – Four Layers of IoT Components

Application Services	Services	Analyze and Learn	Respond
Cloud Computing	Store	Process	Share
Sensor Network	-	Capture	Transmit
Physical Layer	-	Hardware	Software

IoT for PSS

There are few studies, which addressed the relationship between IoT and PSS. Predominantly, the studies investigated application of IoT for PSS development in case studies. For example, Seregni et al. (2016) analyzed three commercial PSS cases, which incorporated IoT technologies into their systems. Based on available public information about the cases, they compared which new services IoT enabled for the PSS cases. They analyzed the cases with regard to four categories, namely, identity-related services, information aggregation services, collaborative-aware services and ubiquity services. Moreover, they investigated whether IoT supported delivery or order phase of the PSS and whether the customer side or the PSS provider side. Nevertheless, the study does not dive deep into the subject and only presents a preliminary analysis.

Another cases study is conducted by Elia et al. (2016) on integrating IoT in a PSS solution for waste collection. The main contribution is the performance evaluation of such a solution and comparing it to traditional non-PSS solutions. The study shows that IoT-enabled PSS is significantly better than traditional methods for waste collection; however, the study rarely focuses on IoT aspects and does not expose any IoT integration insights.

Zancul et al. (2016) propose a method for adopting IoT-enabled PSS regarding its business model. Their method consists of two parts. First, they follow the failure mode and effect analysis (FMEA) approach in order to analyze which features of IoT should be integrated with the product. Second, they use a PSS business strategy configurator that assists PSS providers to positions themselves during the innovation planning. They merge the results of configurator with FMEA approach and determine what product features and PSS processes must be implemented with the help of IoT. They apply and evaluate their method in a case study.

Similarly, Shih et al. (2016) propose a PSS design method that extends visual mapping methods for service creation such as (Matzen and McAloone 2009) and (Moritz 2009) with the aim of incorporating IoT technologies. They introduce a new concept called “pseudo-actor”, which stands for an IoT-enabled object with sensors and actuators. Their method tackles selecting IoT technology alternatives for customer value creation. The method mostly focuses on design of PSS for engineers and the study does not cover general IoT potentials for PSS.

In summary, the existing studies on the integration of IoT in PSS are mostly application-oriented and partially cover the ways, in which IoT support PSS. Particularly, there is lack of knowledge on what general opportunities IoT can provide for PSS in general. Hence, in this study, we build a first theoretical framework to integrate different views on the opportunities of IoT for PSS.

Study Design

To gain a deeper understanding of opportunities of IoT for PSS from a theoretical and a practical perspective, we conducted a structured literature review based on Vom Brocke et al. (2009) and Webster and Watson (2002b) and expert interviews based on Gläser and Laudel (2010), Mayring (2010) and Miles and Huberman (1994). We employed such a mixed-method approach with the purpose of ‘completeness’ (Venkatesh et al. 2013). We aimed to reach a complete picture of the phenomenon of interest by mixing evidences from the literature and practice.

Systematic Literature Review

To analyze opportunities of IoT for PSS from literature, we applied the approach and instructions based on Vom Brocke et al. (2009) and Webster and Watson (2002b). Table 3 gives an overview of this process and resulting numbers of analyzed publications. We used the databases IEEE, SpringerLink, ScienceDirect and Scopus. We applied the following research string: (Lifecycle OR Life-cycle OR “Life cycle”) AND (Development OR Manufacturing OR Production OR Deployment) AND (Interdisciplinary OR Multidisciplinary OR “Product Service System” OR “Cyber Physical System”) OR IoT OR “Internet of Things” OR Servitization OR Digitalization. We included all types of scientific literature and did not confine to a specific publication year range or ranking.

For the analysis, we first analyzed title and abstracts and removed duplicates. We selected only relevant publications based on sets of inclusion and exclusion criteria. The exclusion criteria consists of papers with main focus on IoT implementation or tools. The inclusion criteria are papers, which addressed lifecycle management in the context of IoT and PSS, and IoT integration in business. This selection reduced the number of possible relevant publications to 160. In the second screening, we studied the full text of the papers and evaluated their relevance to our research question. We ended up with 72 relevant papers.

Table 3. Outcome of Database Search

Database	*Initial search*	*Title and abstract screening*	*Full text screening*
IEEE	124	25	17
SpringerLink	1127	72	20
ScienceDirect	53	21	16
Scopus	683	42	19
Total	1987	160	72

Expert Interviews

As the literature review reveals some gaps, we enriched our data with expert interviews based on Gläser and Laudel (2010), Mayring (2010). For the sampling of interviews, we looked for enterprises and start-ups across different application fields of IoT. We chose business managers that consider or involve IoT in their processes, consultants that offer IoT solutions, and start-ups working in the field of IoT. We conducted 13 semi-structured interviews, which their details are presented in Table 4.

Table 4. Interview Details

Interview ID	*Job Description*	*Industry*	*Employees*	*Duration (Minutes)*
Participant 01	Business development manager	Global e-commerce & cloud computing	~566 000	~35
Participant 02	IoT evangelist & business development manager	Global e-commerce & cloud computing	~566 000	~15
Participant 03	Machine Learning Expert	Research institute	~200	~40
Participant 04	Data scientist for rail transportation	Industrial manufacturing	~372 000	~10
Participant 05	Hardware product developer	Start-up in the field of automatization solutions	~12	~20
Participant 06	Innovation manager	Manufacturer of braking systems for rail and commercial vehicles	~25 000	~35
Participant 07	Chief Technology Officer	Start-up in the field of digital gastronomy	~12	~45
Participant 08	Consultant for innovation & product lifecycle management	Global IT consultancy	~120 000	~50
Participant 09	Product manager for digital lab and smart home	Global automotive manufacturer	~125 000	~25
Participant 10	Digital E-Care	Global telecommunication company	~ 1800	~70

Participant 11	IoT consultant and app developer	IoT consultancy and software house	~ 124 000	~35
Participant 12	Product manager for industrial communication	Industrial manufacturing company	~372 000	~20
Participant 13	Consultant and developer	IoT consultancy	~10 000	~50

The interviews were based on a semi-structured interview guideline with open questions (Gläser and Laudel 2010) to ensure some common topics and leave room for specific aspects of every expert. Every expert was asked about general opportunities of IoT and realized applications of IoT (I), opportunities and realized applications resulting from new data (II), and opportunities and realized applications for their specific processes, products or product service systems (III). For data analysis, all interviews were transcribed and openly coded according to Corbin et al. (2014). Our coding is shaped around two core concepts, IoT enablement for PSS business model and IoT enablement for PSS implementation.

IoT as PSS Business Model Enabler

Table 5. Framework of IoT-PSS Business Model Opportunities

		Product-oriented PSS	**Use-oriented PSS**	**Result-oriented PSS**
IoT-Driven PSS	**Transforming**	Autonomous Product and Manufacturing	Continuously Improving Advanced Services	Proactive Smart Results
	Optimizing	Efficient Product and Manufacturing	Personalized Services	Smart Results
IoT-Supported PSS	**Interacting**	Smart Product	Engaging Services	Engaging Results
	Tracking	High Product Quality; Advanced Sales	High Service Quality; Lower Maintenance Cost;	Customized results

As the first part of the results, we present the framework of IoT-PSS business model opportunities (depicted in Table 5). The horizontal axis of the framework stands for three general types of PSS introduced by Tukker (2004). The vertical axis presents the levels of IoT involvement in PSS concept. The four levels are inspired by capability levels of smart products introduced by Porter and Heppelmann (2014) and cover a wide range of IoT implications from simple sensor-enabled products to complex product and service connectivity with autonomous behaviors. The first two levels, namely tracking and interacting, enable IoT-supported PSS. The other levels, namely, optimizing and transforming, enable IoT-driven PSS. While an IoT-supported PSS is a PSS enhanced with IoT technologies, IoT fundamentally affect PSS design and implementation in an IoT-driven PSS. In other words, IoT is the main value creator in an IoT-driven PSS. The inner text of every cell in the framework encapsulates the potential added-value by IoT for each PSS type. However, the value can be derived from many aspects, which we will discuss in this section.

Tracking

Tracking is the lowest level of IoT integration in PSS business models. It enables tracking primary product, service, user and their attributes such as quality and performance metrics. The tracking capability increases awareness of not only the system, but also the environment, in which the PSS is functioning (Lee et al. 2013). For instance, we can even track complex parameters such as frost risk and humidity using smart water sensors (Participant 13). Therefore, the provider would be able to add extra value by improving the quality in use for the users and decrease the maintenance costs (Beuren et al. 2016; Zancul et al. 2016). An important implication of tracking is reflected in product delivery phase and logistics (Barbosa et al. 2016; Papakostas et al. 2016; Porter and Heppelmann 2014). An example of result-oriented PSS enabled by IoT is a wireless connected single-function button that allows customers to order products or services (Participant 02; Participant 01). Tracking and storing processes into these buttons enable us to ask for the result instantly by a click of the button. Another example would be location-based services to users, which is enabled by tracking capabilities of IoT. Therefore, we would be able to improve the customer experience and increase the usage or purchase rate (Participant 07).

Interacting

As the next level, IoT enables a PSS to not only track and report PSS-related data, but also have some degree of action. This can be realized using an event-based scheme or direct interaction with the user. For example, in case of a smart home PSS – in which the home devices and appliances are owned by the PSS provider and the usage is sold to the customer - the lights of a smart home can be turned on or off automatically due to the outside light or the user can directly control them remotely. Similarly, the product would be able to react proactively to a particular condition. The idea is that the product has some degree of self-diagnosis and is able to interact with the user or provider. For example, the user will be informed to replace a part in case of an error. Such an ability increases the customer engagement with the PSS (Participant 13). In general, according to the interviews, interacting capabilities of IoT allows PSS providers to introduce new field services (Participant 04; Participant 06; Participant 07). Connected devices, simple interaction abilities with the environment and conditional clauses – provided by IoT – realize new advanced services for a PSS (Participant 10; Participant 03; Participant 04).

Optimizing

The interviewees argue, although tracking and interacting capabilities added by IoT support creating new business models, they are not sufficient (Participant 03; Participant 13). Thus, we need to involve IoT more into the development of PSS business models and the next step is optimizing capability, which is built upon the preceding capabilities. The collected and processed data during tracking and interacting allows advanced analysis of products and services, particularly in the usage phase. This empowers PSS providers in order to increase the performance of products and services, decrease their costs and identify new opportunities for extending their business models (Vuppala and Kumar 2014). Optimizing capability allows the smartness of a PSS to be dynamic and to evolve through the lifecycle (Barbosa et al. 2016). For instance, sales services become much more intelligent by analyzing the usage data in an IoT-supported PSS (Herterich et al. 2015; Zancul et al. 2016). In addition, pricing can be continuously be calculated in a real-time manner (Zancul et al. 2016). Interviewees perceived great opportunities based on machine learning algorithms, which are able to improve the system functions continuously (Participant 01; Participant 03; Participant 12). They believed such machine learning techniques combined with connectedness of products and services over a PSS enabled by IoT provides opportunities to automate processes and create advanced solutions (Participant 07). Many interviewees emphasized the importance of optimizing with regard to control of PSS failure behavior (Participant 04; Participant 03).

Transforming

Built on the entire IoT technology stack, transforming capability of IoT for PSS is realized by high level of autonomous operations and seamless communication with other networks (Gigli and Koo 2011; Porter and Heppelmann 2014). Transformation for the smart home example means that the home appliances track their usage, perform analysis and accordingly change their behavior, interact with the user as well as other devices and the PSS provider. Therefore, there is a total connectedness and interaction among the people and machines with the aim of maximizing the products performance and quality of services (Participant 09). The products and service provision as well as the customer's experience can significantly be reshaped by total IoT integration (Participant 01). With regard to the autonomy aspect, edge processing - processing power at the edge of the network – is a key ability. It allows local decision makings for every object in the system by collecting raw sensor data, filtering the data and processing the data at its source by intelligent devices (Barbosa et al. 2016; Haller et al. 2008). During the maintenance phase, the system would be able not only to warn the provider or the user, but also to enable the provider to employ a predictive maintenance scheme as well as a real-time autonomous decision making (Zancul et al. 2016). To create more value, it is necessary to establish a combination of machine-learning methods with real-time and cloud-based infrastructure as well as communication across the system's network (Participant 10; Participant 03).

IoT as PSS Lifecycle Management Enabler

Based on the literature and the interviews, we identified the related core potential concepts, which are presented in Figure 1. IoT involvement leads to an increasing amount of data of the PSS and PSS development. The data can be exploited continuously for production improvement and closed-loop lifecycle management reflects this capability. The second aspect tackles collaboration issues in PSS development, which is inherently challenging because of variety of involved disciplines. IoT supports collaboration by enabling communication among machines and humans. Another implication of IoT for the PSS development is the higher degree of autonomy for the PSS development. In addition to the overall concepts, IoT enables specific technologies and paradigms regarding every phase of PSS development. Regarding the PSS development phases of PSS, we follow general accepted differentiation between beginning of life (BOL), middle of life (MOL) and end of life (EOL) phases. These phases present respectively the design, manufacturing, logistics, use, maintenance, reuse and recycling (Beuren et al. 2016; Terzi et al. 2010). Along these phases, we identified four underlying opportunities, namely, digital twin, smart logistics, predictive maintenance and remanufacturing.

Closed-loop Lifecycle Management (CLLM) stands for ubiquity of product-relevant information at any point in the lifecycle (Wiesner et al. 2015; Wuest et al. 2014). Such omnipresence enables stakeholders to track and manage the data even during the use (Kiritsis 2011). In traditional lifecycle management, considerable amount of relevant data is either lost or acquired with high cost. Consequently, there is a limited visibility of products and services for the PSS provider (Basselot et al. 2017; Igba et al. 2015). IoT tracking capabilities overcome such a challenge by low-cost collecting of relevant data among lifecycles of PSS product parts and PSS services (Basselot et al. 2017). Moreover, incorporating IoT into the PSS development would solve challenge of low interoperability among heterogeneous working units that prevents CLLM realization (Basselot et al. 2017; Igba et al. 2015). The interviews reflected the same argument that with the help of IoT, we would collect and manage PSS-related data necessary for CLLM (Participant 01; Participant 06). PSS providers would be able to increase the quality of their product and services continuously. In addition to tracking status of a product, i.e. product-focused data, Matsas et al. (2017) introduce user-focused data, which reflect only usage information and attributes perceived by the user. Utilizing these two types of data can

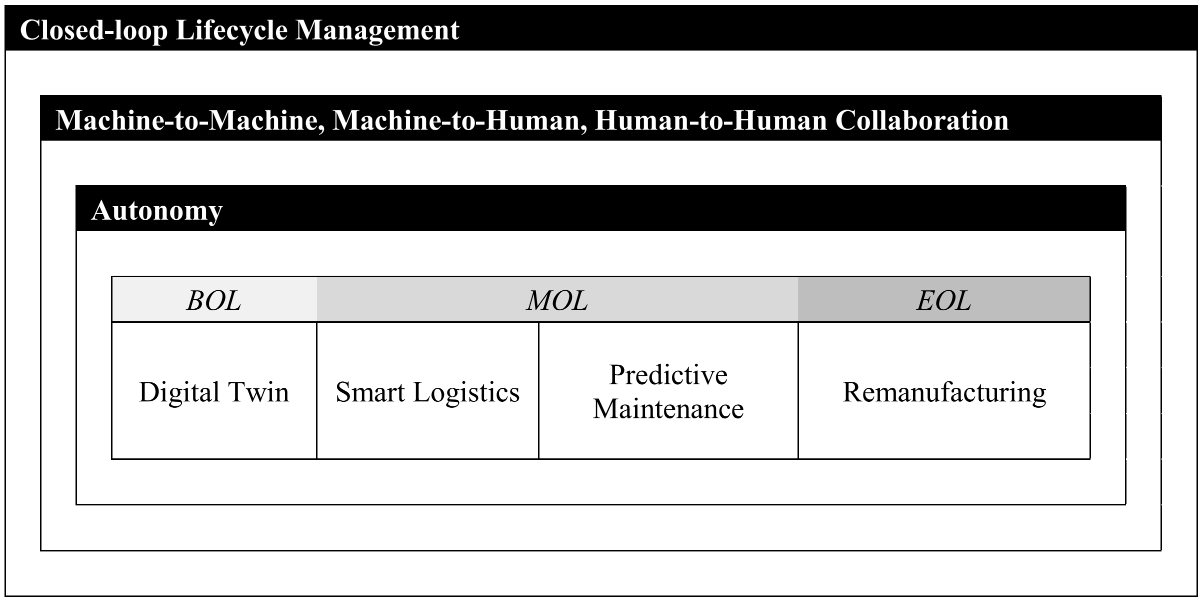

Closed-loop Lifecycle Management
Machine-to-Machine, Machine-to-Human, Human-to-Human Collaboration
Autonomy
BOL
MOL
EOL
Digital Twin
Smart Logistics
Predictive Maintenance
Remanufacturing

Figure 1 – Opportunities of IoT for PSS Lifecycle Management

significantly support requirements elicitation and management for PSS' products and services and even introducing new ones (Gudergan et al. 2017; Wuest and Wellsandt 2016; Yang et al. 2009).

Collaboration-related aspects are challenging for PSS development as PSS development involves high number of teams and disciplines, whose tools and methods differ (Gopsill et al. 2011). IoT capabilities mitigate the severity of such a challenge in collaborations among humans and machines. First, IoT-enhanced machines would be able to transfer their information and adjust their conditions to be aligned with each other. Hence, **Machine-to-Machine (M2M)** collaboration would take place without the human intervention (Lee et al. 2013). With regard to **Human-to-Human (H2H)** collaboration, interviewees from a global e-commerce enterprise highlighted that employing IoT makes the relationship among manufacturers deeper as it increases the interoperability and the supply chain performance can be monitored nearly real-time (Participant 01; Participant 02). Interviewees also agreed that unleashing the potential of a complete IoT solution lead to engagement with new partners, vendors and platforms (Participant 11; Participant 03; Participant 01). Particularly, tools and development platforms in the context of IoT allow a wider range of developers to access the innovative capabilities and build up their knowledge collaboratively (Participant 03). Consequently, companies can focus on their core competence and core business activities (Participant 07).

M2M collaborations enabled by IoT establish new opportunities for process and factory automation by minimizing the human intervention (Ardolino et al. 2016; Gerpott and May 2016; Lee et al. 2013). Interviews showed cases in which IoT could automate the complete supply chain processes from an order on the website to final delivery. This led to cost reduction and improved customer experience (Participant 01; Participant 02). Moreover, incorporating advanced machine learning techniques based on data collected and filtered by IoT empowers **autonomous** decision-makings, self-coordination and self-diagnosis abilities (Porter and Heppelmann 2014), which is confirmed by the interviews (Participant 11; Participant 03). However, the interviewees argued that there are several challenges that impede realizing high autonomy. For example, yet there are no advances in automated self-criticism, in which the system recognizes its mistakes (Participant 03). In addition, there is still lack of trust in automation operations, which does not allow its full integration into the lifecycle management (Participant 03).

Digital twin or product avatar refers to digital equivalent of a physical product. Integrating actual physical data with the virtual replication of a product enables a better design, validation and verification of engineering artefacts (Goto et al. 2016). In general, there is a trend towards use of digital twin enabled by IoT capabilities (Participant 08). Digital twin can be engaged for predicting, optimizing and verifying the products along the lifecycle. However, it plays a significant role in BOL phase by incorporating feedbacks from MOL and EOL phases into improving the design and simulating different options (Participant 01; Participant 02). For instance, a digital presentation of a product supports evaluating performance of the product in diverse environments. Moreover, applying a change in PSS can be first reflected in the virtual setting and the results can be used to realize PSS more efficiently (Participant 02; Participant 08). Another important ability of digital twin is that we can present the system thoroughly and more easily to different stakeholders along the entire lifecycle (Participant 02; Participant 08). Use of digital twin reduces the delays, increases the overall development efficiency and transparency of customers' processes (Meneghetti et al. 2016).

Smart logistics is enabled by tracking and optimizing abilities of IoT. IoT establishes an overall connectivity of all devices and product parts, which empowers efficient delivery of products and integrated services (Vuppala and Kumar 2014). For instance, IoT supports activities such as resource allocation (Barbosa et al. 2016) and inventory management (Papakostas et al. 2016). Moreover, with the help of IoT, autonomous vehicles would be able to optimize transportations during the manufacturing and facilitate distributed orders (Mueller et al. 2017). Based on the interviews, such capabilities of IoT are currently in use in several manufacturing leaders (Participant 01).

Predictive maintenance is regular monitoring and analyzing of the system conditions in order to minimize the number of failures and repairs (Mobley 2002). Since IoT provides valuable insight with regard to the PSS and its usage, it can minimize the time for error diagnosis (Lerch and Gotsch 2015). For example, with the help of IoT sensors and analysis of the collected usage data, we would be able to elicit spare part requirements (Herterich et al. 2015; Zancul et al. 2016) Several interviewees reported that they have experienced considerable savings by incorporating IoT capabilities into the maintenance activities (Participant 01; Participant 08; Participant 05). Moreover, they stated that increased availability resulted from a more efficient maintenance led to higher customer satisfaction.

Remanufacturing stands for the industrial process, in which we restore and recover used products into a good condition (Lindkvist and Sundin 2016). Hence, experiences of later stages of lifecycle would be employed in the earlier stages (Igba et al. 2015). Realizing remanufacturing necessitates tracking, controlling and analyzing the product, the condition of the product and the usage of product, which can be enabled by means of IoT (Chierici and Copani 2016). Ideally, there is a feedback loop between each lifecycle phases.

Discussion

IoT paradigm can transform the industry and be as influential as the Internet was in the 1990s. Our findings showed that practitioners assert high potential of IoT for facilitating new business models, designing new products and providing advanced services. In conformance with this fact, the prior research emphasized on transforming abilities of IoT and the big impact that IoT can have on businesses (Čolaković and Hadžialić 2018; Gubbi et al. 2013; Porter and Heppelmann 2014). Particularly, IoT can play a crucial role for PSS (Seregni et al. 2016; Shih et al. 2016; Zancul et al. 2016). Due to challenging nature of PSS, which transforms merely product or service businesses into an integrated enterprise of product and service provision, more connectedness and communication among heterogeneous elements is necessary (Vasantha et al. 2012; Wiesner et al. 2015). Strengths of IoT matches to the difficulties that PSS design and development confront.

The existing studies on IoT and PSS relationship limit to single case applications of a particular method for adopting IoT in PSS development (Shih et al. 2016; Zancul et al. 2016). We extend the current literature by establishing a comprehensive view on the opportunities that IoT can provide for PSS. We presented the framework of IoT-PSS business model opportunities that introduces four levels of IoT involvement in PSS. Based on the framework, there is a wide range of IoT integration into PSS. It starts from basic IoT-supported tracking abilities in PSS to transformed IoT-driven PSS with IoT as its core value creator. The framework assists PSS providers in positioning themselves, identifying the extent, to which they have already benefited from IoT and the possibilities, which they have not realized yet. Furthermore, we identified and highlighted the core IoT-enabled opportunities, which facilitate PSS lifecycle management. Although the concepts vary largely from M2M collaboration to digital twin and remanufacturing, they are mutual in terms of being enabled by IoT and advancing PSS lifecycle management. Nevertheless, diving deep into the details of implementing such technologies in the domain of PSS was out of scope of this study and can be investigated in future research. We argue that our study provides the fundamentals for advancing PSS and IoT integration research. Future studies can build new concepts, methods and tools upon the established frameworks of this study.

Combining the two folds of this study's contribution enlighten the overall IoT exploitation for PSS design and development. Insightful alignment of IoT and PSS allows various added-values for both businesses and the customers. Regarding the customer values, PSS providers would be able to establish a reliable connection with the customer, partners and suppliers by a right IoT integration. Customers can expect a continuous improving product and service, which are also more customized to their usage. In addition, customers would benefit from a higher availability of product and services. In context of the business values, IoT integration shortens the development cycles and reduces costs of development. PSS providers will have a shorter time-to-market, which is a decisive aspect in a competitive environment. Moreover, utilizing IoT decreases costs of maintenance and remanufacturing significantly. For example, there would be no need for on-site monitoring of product conditions as the sensors are continuously tracking the relevant information. At its extreme realization, PSS providers will gain autonomy and transparency during all phases of PSS lifecycle. Even though a limited integration of IoT in PSS enables PSS providers to introduce smart products and advanced services, which can lead to a higher revenue.

According to our findings from the interviews, IoT technologies have been integrated mostly on the end-customers side, despite the fact that B2B applications of IoT can have greater economic outcomes. Moreover, we observed a slow progress regarding the shift from IoT-supported PSS to IoT-driven PSS. Although, lack of infrastructural capabilities can be considered as an important factor that stops IoT integration, complicated barriers exist, which future studies need to investigate them in detail. For example, there is still uncertainty about costs and profits of IoT adoption, particularly at its highest extent. Mechanisms to analyze and estimate IoT adoption in terms of monetary parameters would significantly support the realization of IoT opportunities. Furthermore, IoT integration is fostering a collaborative ecosystem, in which many start-ups have emerged as IoT technology providers. Future studies can look more into how we can ease the integration of such start-ups' contributions into existing infrastructures. With this regard, the research should study the role of emerging IoT platforms, which will facilitate use of IoT for variety of applications.

Conclusion

In addition to empowering the existing solutions, IoT enables us to realize new ideas. Particularly, we can use the power of IoT to facilitate complexity of PSS design and development. In this study, we investigated opportunities that IoT can provide for PSS business models and lifecycle management. We provided examples for each relevant hotspot to assist PSS providers in positioning and deciding a right business model when integrating IoT in their portfolio. First, we introduced framework of IoT opportunities for PSS business models that entails two dimensions of IoT involvement level and PSS types. It evaluates which type of services IoT technologies foster for the provision of PSS. Furthermore, we analyzed IoT as a key facilitator of the lifecycle management by enabling new technologies and capabilities such as autonomy, closed-lifecycle management, digital twin, predictive maintenance and remanufacturing.

The findings of this study provide new insights for PSS providers. Moreover, this study establishes a comprehensive view on opportunistic implications of IoT for PSS, which paves the path for future studies to advance this topic. The research can complete this work by addressing on one hand, the barriers for integrating IoT into PSS and on the other hand, the challenges caused by IoT integration into PSS. Accordingly, the studies can propose solutions to overcome such challenges.

Acknowledgements

This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) collaborative research centre 'Sonderforschungsbereich SFB 768 "Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products".

References

Adrodegari, F., and Saccani, N. 2017. "Business Models for the Service Transformation of Industrial Firms," The Service Industries Journal (37:1), pp. 57-83.

Ardolino, M., Saccani, N., Gaiardelli, P., and Rapaccini, M. 2016. "Exploring the Key Enabling Role of Digital Technologies for Pss Offerings," Procedia CIRP (47), pp. 561-566.

Baines, T. S., Lightfoot, H. W., Evans, S., Neely, A., Greenough, R., Peppard, J., Roy, R., Shehab, E., Braganza, A., and Tiwari, A. 2007. "State-of-the-Art in Product-Service Systems," Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture (221:10), pp. 1543-1552.

Bandyopadhyay, D., and Sen, J. 2011. "Internet of Things: Applications and Challenges in Technology and Standardization," Wireless Personal Communications (58:1), pp. 49-69.

Barbosa, J., Leitão, P., Trentesaux, D., Colombo, A. W., and Karnouskos, S. 2016. "Cross Benefits from Cyber-Physical Systems and Intelligent Products for Future Smart Industries," Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on: IEEE, pp. 504-509.

Basirati, M. R., Zou, M., Bauer, H., Kattner, N., Reinhart, G., Lindemann, U., Böhm, M., Krcmar, H., and Vogel-Heuser, B. 2018. "Towards Systematic Inconsistency Identification for Product

Service Systems," DS92: Proceedings of the DESIGN 2018 15th International Design Conference, pp. 2811-2820.

Basselot, V., Berger, T., and Sallez, Y. 2017. "Active Monitoring of a Product: A Way to Solve the "Lack of Information" Issue in the Use Phase," in Service Orientation in Holonic and Multi-Agent Manufacturing. Springer, pp. 337-346.

Beuren, F. H., Ferreira, M. G. G., and Miguel, P. A. C. 2013. "Product-Service Systems: A Literature Review on Integrated Products and Services," Journal of cleaner production (47), pp. 222-231.

Beuren, F. H., Pereira, D., and Fagundes, A. B. 2016. "Product-Service Systems Characterization Based on Life Cycle: Application in a Real Situation," Procedia CIRP (47), pp. 418-423.

Chierici, E., and Copani, G. 2016. "Remanufacturing with Upgrade Pss for New Sustainable Business Models," Procedia CIRP (47), pp. 531-536.

Čolaković, A., and Hadžialić, M. 2018. "Internet of Things (Iot): A Review of Enabling Technologies, Challenges, and Open Research Issues," Computer Networks).

Corbin, J., Strauss, A., and Strauss, A. L. 2014. Basics of Qualitative Research. sage.

Elia, V., Gnoni, M. G., and Tornese, F. 2016. "Assessing the Efficiency of a Pss Solution for Waste Collection: A Simulation Based Approach," Procedia CIRP (47), pp. 252-257.

Exner, K., Zimpfer, R., and Stark, R. 2017. "Maturity Model and Action Recommendation: A Pss Capability Self-Assessment Tool for Companies," Procedia CIRP (64), pp. 175-180.

Georgakopoulos, D., and Jayaraman, P. P. 2016. "Internet of Things: From Internet Scale Sensing to Smart Services," Computing (98:10), pp. 1041-1058.

Gerpott, T. J., and May, S. 2016. "Integration of Internet of Things Components into a Firm's Offering Portfolio–a Business Development Framework," info (18:2), pp. 53-63.

Gigli, M., and Koo, S. G. 2011. "Internet of Things: Services and Applications Categorization," Adv. Internet of Things (1:2), pp. 27-31.

Gläser, J., and Laudel, G. 2010. Experteninterviews Und Qualitative Inhaltsanalyse. Springer-Verlag.

Goedkoop, M. J., Van Halen, C. J., Te Riele, H. R., and Rommens, P. J. 1999. "Product Service Systems, Ecological and Economic Basics," Report for Dutch Ministries of environment (VROM) and economic affairs (EZ) (36:1), pp. 1-122.

Gopsill, J. A., McAlpine, H. C., and Hicks, B. J. 2011. "Learning from the Lifecycle: The Capabilities and Limitations of Current Product Lifecycle Practice and Systems," DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6: Design Information and Knowledge, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011.

Goto, S., Yoshie, O., and Fujimura, S. 2016. "Internet of Things Value for Mechanical Engineers and Evolving Commercial Product Lifecycle Management System," Industrial Engineering and Engineering Management (IEEM), 2016 IEEE International Conference on: IEEE, pp. 1021-1024.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. 2013. "Internet of Things (Iot): A Vision, Architectural Elements, and Future Directions," Future generation computer systems (29:7), pp. 1645-1660.

Gudergan, G., Buschmeyer, A., Feige, B. A., Krechting, D., Bradenbrink, S., and Mutschler, R. 2017. "Value of Lifecycle Information to Transform the Manufacturing Industry," in Shaping the Digital Enterprise. Springer, pp. 173-194.

Guth, J., Breitenbücher, U., Falkenthal, M., Fremantle, P., Kopp, O., Leymann, F., and Reinfurt, L. 2018. "A Detailed Analysis of Iot Platform Architectures: Concepts, Similarities, and Differences," in Internet of Everything. Springer, pp. 81-101.

Haller, S., Karnouskos, S., and Schroth, C. 2008. "The Internet of Things in an Enterprise Context," Future Internet Symposium: Springer, pp. 14-28.

Herterich, M. M., Uebernickel, F., and Brenner, W. 2015. "The Impact of Cyber-Physical Systems on Industrial Services in Manufacturing," Procedia CIRP (30), pp. 323-328.

Igba, J., Alemzadeh, K., Gibbons, P. M., and Henningsen, K. 2015. "A Framework for Optimising Product Performance through Feedback and Reuse of in-Service Experience," Robotics and Computer-Integrated Manufacturing (36), pp. 2-12.

Kiel, D., Arnold, C., and Voigt, K.-I. 2017. "The Influence of the Industrial Internet of Things on Business Models of Established Manufacturing Companies–a Business Level Perspective," Technovation (68), pp. 4-19.

Kiritsis, D. 2011. "Closed-Loop Plm for Intelligent Products in the Era of the Internet of Things," Computer-Aided Design (43:5), pp. 479-501.

Kowalkowski, C., Windahl, C., Kindström, D., and Gebauer, H. 2015. "What Service Transition? Rethinking Established Assumptions About Manufacturers' Service-Led Growth Strategies," Industrial Marketing Management (45), pp. 59-69.

Lee, G. M., Crespi, N., Choi, J. K., and Boussard, M. 2013. "Internet of Things," in Evolution of Telecommunication Services. Springer, pp. 257-282.

Lerch, C., and Gotsch, M. 2015. "Digitalized Product-Service Systems in Manufacturing Firms: A Case Study Analysis," Research-Technology Management (58:5), pp. 45-52.

Lightfoot, H., Baines, T., and Smart, P. 2013. "The Servitization of Manufacturing: A Systematic Literature Review of Interdependent Trends," International Journal of Operations & Production Management (33:11/12), pp. 1408-1434.

Lindkvist, L., and Sundin, E. 2016. "The Role of Product-Service Systems Regarding Information Feedback Transfer in the Product Life-Cycle Including Remanufacturing," Procedia Cirp (47), pp. 311-316.

Maleki, E., Belkadi, F., Zhang, Y., and Bernard, A. 2017. "Towards a New Collaborative Framework Supporting the Design Process of Industrial Product Service Systems," in Advances on Mechanics, Design Engineering and Manufacturing. Springer, pp. 139-146.

Marilungo, E., Papetti, A., Germani, M., and Peruzzini, M. 2017. "From Pss to Cps Design: A Real Industrial Use Case toward Industry 4.0," Procedia CIRP (64), pp. 357-362.

Matsas, M., Pintzos, G., Kapnia, A., and Mourtzis, D. 2017. "An Integrated Collaborative Platform for Managing Product-Service across Their Life Cycle," Procedia CIRP (59), pp. 220-226.

Matzen, D., and McAloone, T. C. 2009. A Systematic Apporach to Service Oriented Product Development. DTU Management.

Maxwell, D., Sheate, W., and Van Der Vorst, R. 2006. "Functional and Systems Aspects of the Sustainable Product and Service Development Approach for Industry," Journal of cleaner production (14:17), pp. 1466-1479.

Mayring, P. 2010. "Qualitative Inhaltsanalyse," in Handbuch Qualitative Forschung in Der Psychologie. Springer, pp. 601-613.

Mazhelis, O., Luoma, E., and Warma, H. 2012. "Defining an Internet-of-Things Ecosystem," in Internet of Things, Smart Spaces, and Next Generation Networking. Springer, pp. 1-14.

Meier, H., Roy, R., and Seliger, G. 2010. "Industrial Product-Service Systems—Ips2," CIRP Annals-Manufacturing Technology (59:2), pp. 607-627.

Meneghetti, A., Moro, S., and Helo, P. 2016. "Intermixed Product and Service Boundaries: Exploring Servitization in Sheet Metal Industry," Procedia CIRP (47), pp. 258-263.

Miles, M. B., and Huberman, A. M. 1994. Qualitative Data Analysis: An Expanded Sourcebook. sage.

Mobley, R. K. 2002. An Introduction to Predictive Maintenance. Elsevier.

Mont, O. K. 2002. "Clarifying the Concept of Product–Service System," Journal of cleaner production (10:3), pp. 237-245.

Moritz, S. 2009. Service Design: Practical Access to an Evolving Field. Lulu. com.

Mueller, E., Chen, X.-L., and Riedel, R. 2017. "Challenges and Requirements for the Application of Industry 4.0: A Special Insight with the Usage of Cyber-Physical System," Chinese Journal of Mechanical Engineering (30:5), pp. 1050-1057.

Papakostas, N., O'Connor, J., and Byrne, G. 2016. "Internet of Things Technologies in Manufacturing: Application Areas, Challenges and Outlook," Information Society (i-Society), 2016 International Conference on: IEEE, pp. 126-131.

Patel, K. K., and Patel, S. M. 2016. "Internet of Things-Iot: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," International journal of engineering science and computing (6:5).

Porter, M. E., and Heppelmann, J. E. 2014. "How Smart, Connected Products Are Transforming Competition," Harvard business review (92:11), pp. 64-88.

Reim, W., Parida, V., and Örtqvist, D. 2015. "Product–Service Systems (Pss) Business Models and Tactics – a Systematic Literature Review," Journal of Cleaner Production (97), pp. 61-75.

Schuh, G., Salmen, M., Kuhlmann, T., and Wiese, J. 2016. "Life-Cycle-Oriented Product-Service-Systems in the Tool and Die Making Industry," Procedia CIRP (47), pp. 555-560.

Seregni, M., Sassanelli, C., Cerri, D., Zanetti, C., and Terzi, S. 2016. "The Impact of Iot Technologies on Product-Oriented Pss: The "Home Delivery" Service Case," Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on: IEEE, pp. 1-5.

Shih, L.-H., Lee, Y.-T., and Huarng, F. 2016. "Creating Customer Value for Product Service Systems by Incorporating Internet of Things Technology," Sustainability (8:12), p. 1217.

Tao, F., Zuo, Y., Da Xu, L., and Zhang, L. 2014. "Iot-Based Intelligent Perception and Access of Manufacturing Resource toward Cloud Manufacturing," IEEE Transactions on Industrial Informatics (10:2), pp. 1547-1557.

Terzi, S., Bouras, A., Dutta, D., Garetti, M., and Kiritsis, D. 2010. "Product Lifecycle Management-from Its History to Its New Role," International Journal of Product Lifecycle Management (4:4), pp. 360-389.

Tukker, A. 2004. "Eight Types of Product–Service System: Eight Ways to Sustainability? Experiences from Suspronet," Business strategy and the environment (13:4), pp. 246-260.

Ulaga, W., and Reinartz, W. J. 2011. "Hybrid Offerings: How Manufacturing Firms Combine Goods and Services Successfully," Journal of marketing (75:6), pp. 5-23.

Vasantha, G. V. A., Roy, R., Lelah, A., and Brissaud, D. 2012. "A Review of Product–Service Systems Design Methodologies," Journal of Engineering Design (23:9), pp. 635-659.

Venkatesh, V., Brown, S. A., and Bala, H. 2013. "Bridging the Qualitative-Quantitative Divide: Guidelines for Conducting Mixed Methods Research in Information Systems," MIS quarterly), pp. 21-54.

Vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., and Cleven, A. 2009. "Reconstructing the Giant: On the Importance of Rigour in Documenting the Literature Search Process," ECIS, pp. 2206-2217.

Vuppala, S. K., and Kumar, H. K. 2014. "Service Applications-Exploiting the Internet of Things," Global Conference (SRII), 2014 Annual SRII: IEEE, pp. 195-202.

Webster, J., and Watson, R. T. 2002a. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," MIS Quarterly (26:2), pp. xiii-xxiii.

Webster, J., and Watson, R. T. 2002b. "Analyzing the Past to Prepare for the Future: Writing a Literature Review," MIS quarterly), pp. xiii-xxiii.

Weking, J., Brosig, C., Böhm, M., Hein, A., and Krcmar, H. 2018. "Business Model Innovation Strategies for Product Service Systems–an Explorative Study in the Manufacturing Industry," Twenty-Sixth European Conference on Information Systems (ECIS 2018).

Whitmore, A., Agarwal, A., and Da Xu, L. 2015. "The Internet of Things—a Survey of Topics and Trends," Information Systems Frontiers (17:2), pp. 261-274.

Wiesner, S., Freitag, M., Westphal, I., and Thoben, K.-D. 2015. "Interactions between Service and Product Lifecycle Management," Procedia CIRP (30), pp. 36-41.

Wortmann, F., and Flüchter, K. 2015. "Internet of Things," Business & Information Systems Engineering (57:3), pp. 221-224.

Wuest, T., Hribernik, K., and Thoben, K.-D. 2014. "Capturing, Managing and Sharing Product Information Along the Lifecycle for Design Improvement," Proceedings of the 10th International Workshop on Integrated Design Engineering.

Wuest, T., and Wellsandt, S. 2016. "Design and Development of Product Service Systems (Pss)-Impact on Product Lifecycle Perspective," Procedia Technology (26), pp. 152-161.

Yang, X., Moore, P., Pu, J.-S., and Wong, C.-B. 2009. "A Practical Methodology for Realizing Product Service Systems for Consumer Products," Computers & Industrial Engineering (56:1), pp. 224-235.

Zancul, E. d. S., Takey, S. M., Barquet, A. P. B., Kuwabara, L. H., Cauchick Miguel, P. A., and Rozenfeld, H. 2016. "Business Process Support for Iot Based Product-Service Systems (Pss)," Business Process Management Journal (22:2), pp. 305-323.

Not Included in Review and Evaluation: P2

Asia Pacific Journal of Information Systems
Vol. 29 No. 3 (September 2019), 524-546

ISSN 2288-5404 (Print) / ISSN 2288-6818 (Online)
https://doi.org/10.14329/apjis.2019.29.3.524

Special Issue on Smart Services and Internet of Things

Exploring Opportunities of IoT for Product-Service System Conceptualization and Implementation

Mohammad R. Basirati[a,*], Jörg Weking[b], Sebastian Hermes[c], Markus Böhm[d], Helmut Krcmar[e]

[a] *Research associate and Ph.D. student, Chair for Information Systems, Technical University of Munich (TUM), Germany*
[b] *Research associate and Ph.D. student, Chair for Information Systems, Technical University of Munich (TUM), Germany*
[c] *Research associate and Ph.D. student, Chair for Information Systems, Technical University of Munich (TUM), Germany*
[d] *Research Group Leader, Chair for Information Systems, Technical University of Munich (TUM), Germany*
[e] *Chair Professor, Information Systems, Technical University of Munich (TUM), Germany*

ABSTRACT

Product-service systems (PSS), integrating physical products and services, currently play a crucial role in sustainable economies. In addition to the highly competitive global economy, the emergence of new digital paradigms is supporting the shift toward servitization. Although the great potential of such paradigms is recognized by both practice and research, their implications for PSS are not yet clear. In particular, features of Internet of Things (IoT), such as total connectedness and ubiquity of smart sensors and actuators, provide various new opportunities for PSS. This study explores such opportunities by conducting structured literature review and 13 interviews. We organize the findings in two folds: First, we introduce four degrees of IoT involvement in PSS business models and elaborate the opportunities that they create for different types of PSS. Second, we present the key technologies and approaches that IoT provides concerning PSS lifecycle management.

Keywords: Product-Service System, Internet of Things, IoT Integration, Review, Expert Interview

Ⅰ. Introduction

Firms have to increase their share of service offerings in order to survive in today's competitive global economy (Mont, 2002). Products are no longer the main contributors to value creation, as the value is shifting toward services. We can see this shift in the gross domestic product of most developed countries, which are more dependent on services than physical products (Meier et al., 2010). Consequently, more service-oriented business models, such as product - service systems (PSS), have emerged. Most definitions of PSS describe it as a system that integrates products and services in order to create a competitive

*Corresponding Author. E-mail: mohammadreza.basirati@tum.de Tel: 498928919598

solution (Beuren et al., 2013), while some definitions also emphasize its role in reaching sustainability with regard to environmental and social considerations (Baines et al., 2007; Maxwell et al., 2006).

A frequently cited example of a PSS is that of the Xerox company (Baines et al., 2007). Traditionally manufacturing print and copy machines, now Xerox provides print and copy solutions that comprise more service-side than product-side elements. A starting point for such a change was developing a pay-per-copy system, in which the machines were sold at a low price and copy function was seen as a service provided by Xerox. More recent and widespread examples of PSS are car-, bike-, and e-scooter-sharing systems. In such a PSS, the value is created by providing mobility as a service to the customer instead of selling a physical product such as a car.

At the same time, the emergence of advanced digital paradigms such as the Internet of Things (IoT) is providing even more opportunities for innovative service offerings and PSS design (Kowalkowski et al., 2015; Lightfoot et al., 2013; Ulaga and Reinartz, 2011). IoT is a concept describing the networking of objects, which can sense, communicate, store data, and interact with the environment (Patel and Patel, 2016). IoT allows not only monitoring the status of physical objects but also establishes the basis for progressive services such as optimization and automization of product operations and services (Adrodegari and Saccani, 2017; Porter and Heppelmann, 2014).

There is a consensus among previous studies on the relevance of digital technologies such as IoT to servitization, particularly PSS (Exner et al., 2017; Marilungo et al., 2017; Shih et al., 2016). In practice, however, the adoption of IoT is a challenging issue, as it requires an intensive reconfiguration of existing settings (Marilungo et al., 2017). However, past research has not provided clear guidance on how we can exploit IoT to successfully design and develop PSS despite the need (Kiel et al., 2017). Hence, this study addresses the following research question:

What opportunities does IoT provide for PSS design and development?

We focus on two important aspects of PSS development: (1) integrating IoT into PSS business models, and (2) integrating IoT in the PSS lifecycle. To get a wide-ranging understanding of IoT and PSS in research and practice, we use a structured literature review (Webster and Watson, 2002) and interview experts (Gläser and Laudel, 2010; Miles and Huberman, 1994). The results provide a comprehensive overview of ideas and practices that IoT delivers for innovative PSS design and development. With regard to the business-development aspect, a framework elaborates the implications of different degrees of IoT involvement for different types of PSS. We also present the core concepts of IoT and the technologies it enables, which can be employed to facilitate PSS lifecycle management. We extend a previously published study (Basirati et al., 2019) in two steps: First, we cover related prior research more comprehensively, and second, we provide real-world PSS cases for every aspect of IoT opportunity for PSS business models.

Ⅱ. Conceptual Background

2.1. Product - Service Systems

PSS refers to a strategic business-model design intended to integrate and combine products, services, and communication based on changing customer and stakeholder demands (Beuren et al., 2013). The concept was introduced in 1999 as a promising business model for "sustainable economic growth" (Baines et al., 2007; Maleki et al., 2017). Most articles investigating PSS rely on definitions by Goedkoop et al. (1999):

"A product–service system is a marketable set of products and services capable of jointly fulfilling a user's need. A product is a tangible commodity, manufactured to be sold. A service is an activity (work), often done on a commercial basis and for others, with an economic value. A system is a combination of elements including their relations."

Recognizing the importance of implementing integrated product–service offerings, PSS literature has considered them a powerful source of competitive advantage and sustainability (Ardolino et al., 2016; Schuh et al., 2016). PSS has proven to provide advantages such as higher profit margins, new growth opportunities in saturated markets, and long-term customer relationships. Besides the advantages for PSS providers, PSS also benefits consumers, the environment, and society (Beuren et al., 2013). Nonetheless, PSS implementation can be challenging and lead to inconsistencies among heterogeneous teams and developing artifact (Basirati et al., 2018). Moreover, PSS adoption into existing business models is not a straightforward procedure and requires applying proper strategies (Weking et al., 2018).

Within the PSS research stream, three types of PSS have emerged: product-oriented, use-oriented, and result-oriented PSS (Baines et al., 2007; Tukker, 2004; Yang et al., 2009). This classification is widely accepted in the literature. <Table 1> describes the three different categories of PSS in terms of their underlying business-model elements (Reim et al., 2015).

Another way of looking at the three types of PSS is to consider what point they have reached on the innovation scale; result-oriented PSS is the most innovative, and product-oriented PSS is the least innovative. For a PSS to evolve from product-oriented to result-oriented, it may take incremental steps and/or a radical path. Incremental innovation, in this context, means that product-oriented PSS evolves slowly to use-oriented and then further to result-oriented. This happens through a slow and steady continuous-improvement process. Radical innovation, on the other hand, means that product-oriented PSS transforms directly into result-oriented PSS, skipping the use-oriented stage. This often involves a radical shift in technology and leads to a total reconfiguration of the PSS.

The Xerox case, introduced in the previous section, is a typical product-oriented PSS example. All manu-

<Table 1> PSS Types According to Reim et al. (2015)

	Product-oriented	Use-oriented	Result-oriented
Value creation	The main responsibility of provider is service delivery.	The main responsibility of provider is the usability of the product or service.	Results are the main responsibility of provider.
Value delivery	Provider delivers extra services in addition to sold products (e.g., maintenance or recycling).	Provider focuses on service usability along with product usability.	The results are counted as the main deliverables instead of products or services.
Value capturing	Customer pays for the product and extra delivered services.	The payment is performed over usage phase continuously (e.g., leasing).	Customer pays based on outcome units instead of pay-per-use or pay-per-product.

facturers that provide maintenance and recycling services besides their products can be considered examples of the product-oriented PSS type.

Car-sharing and bike-sharing cases belong to use-oriented PSS type. In such cases, the price is calculated based on the units of usage. For example, BMW car-sharing service DriveNow1) and Daimler car-sharing service car2go2) charge the users based on a per-minute basis. The users can take any available car in the city and park it for free anywhere in the city. The cars (physical products) are typical car models manufactured by BMW and Daimler. However, these car manufacturers do not sell their cars in the PSS, but they use their physical products as a means to deliver mobility services to the users.

Result-oriented PSS has the highest level of servitization, in which the service-side creates more share of value than the product side (Yang et al., 2010). If a washing machine manufacturer provides its machines for free and charges the users on a pay-per-use basis, this would be a use-oriented PSS. It is possible to incorporate more servitization in such a PSS by delivering laundered clothes, i.e. the result, instead of the machines (Baines et al., 2007). Such a system would be a result-oriented PSS. A real-case advanced example of result-oriented PSS is Lufthansa's AVIATAR digital power platform3), which provides various apps and services for airlines and their suppliers and partners. For instance, the airlines can create networks with each other and share their airplanes' spare parts with the purpose of increasing the availability.

1) https://www.drive-now.com

2) https://www.car2go.com

3) https://www.lufthansa-technik.com/aviatar

2.2. Internet of Things

The term "Internet of Things" was introduced by Kevin Ashton in a presentation in 1998 (Perera et al., 2014) and is now a technological concept with widespread applications (Tao et al., 2014). However, there is as yet no standard definition for IoT, as research about IoT is still in its infancy. Building on the seminal work of Gubbi et al. (2013), we define IoT as follows:

"Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large-scale sensing, data analytics, and information representation using cutting-edge ubiquitous sensing and cloud computing."

The IoT concept includes both technology and services that are based on connected objects and the use of the collected data (Čolaković and Hadžialić, 2018). Everyday objects can be equipped with sensors and actuators to communicate, generate, and process data (Whitmore et al., 2015). Usually, an object, also called a thing, communicates over network protocols with a service in the cloud (Guth et al., 2018).

We elaborate the essential components of IoT within a four-layered technology stack (<Table 2>), which we based on the insights of several others earlier in this decade (Bandyopadhyay and Sen, 2011; Georgakopoulos and Jayaraman, 2016; Lee et al., 2013; Mazhelis et al., 2012; Porter and Heppelmann, 2014; Vuppala and Kumar, 2014; Wortmann and Flüchter, 2015). <Table 2> illustrates the multiple technology layers, including the physical, sensor network, cloud computing services and application services layer.

<Table 2> Four Layers of IoT Components

Application services	Analyzing data, learning and responding
Cloud computing	Storing, processing and sharing data
Sensor network	Transmitting data
Physical layer	Providing hardware infrastructure

The four layers are independent, which means that all components can be developed independently. Communication between the components ideally takes place through well-defined interfaces and a cloud-based shared platform. In general, the two lower layers are responsible for data capturing (data generation and collection) by low-end sensor nodes, while the two upper layers contribute to data processing and utilization in applications.

Each of the four layers (<Table 2>) has distinct capabilities, operations, and costs. The physical layer provides the infrastructural hardware components, such as embedded sensors, processors, and data storage. Connectivity among these components is reflected in the sensor-network layer, in which the data are transmitted by various technologies, such as Bluetooth, Ethernet, or RFID. To store and process the huge amounts of data captured in the physical layer and transmitted in the sensor-network layer, the cloud-computing layer provides mechanisms to aggregate and normalize the data. This layer also makes available the main attributes of the data for detailed analysis and may connect external sources of data (e.g., traffic and weather data). This layer can be considered as a platform that supports heterogenous devices, data privacy and security, and total integration within a bigger ecosystem (Marques et al., 2017). In the highest layer in the IoT technology stack, the application-services layer, the unit of analysis and operations is large in scale compared to the cloud-computing layer. On the basis of the other three layers, this layer provides a deep analysis of data and appropriate services.

On the basis of the extent to which each layer is configured and implemented, IoT can provide different PSS opportunities. Although the physical and sensor-network layers exist in every IoT system regardless of configuration, the cloud-computing and service-application layers may be absent. Therefore, the level of realization of each discussed layer can reflect the extent to which IoT can affect PSS design and development.

2.3. Internet of Things for Product–Service Systems

Few studies have addressed the relationship between IoT and PSS. Most are very recent and mainly use case studies to investigate the application of IoT-for-PSS development. For example, Seregni et al. (2016) analyzed three commercial PSS cases that incorporated IoT technology into their systems. Available information about the cases indicates that they compared which new services IoT-enabled for the PSS cases. They analyzed the cases with regard to four categories—identity-related, information-aggregation, collaborative-awareness, and ubiquity services. They also investigated whether IoT-supported the delivery or order phase of the PSS and whether on the customer side or the PSS provider side. Nevertheless, the study does not go into the subject at depth, instead presenting only a preliminary analysis. Another case study, conducted by Elia et al. (2016) looked at integrating IoT in a PSS solution for waste collection; its main contributions are an evaluation of the performance of such a solution and its comparison to traditional non-PSS solutions. The study shows that IoT-enabled PSS is significantly better than traditional methods for waste collection; however, the study rarely focuses on the IoT aspects and does not mention any IoT-integration insights. Bressanelli et al. (2018) conducted an explorative case study to understand how IoT can ease the challenges of PSS development. The case is a household-appliance retailer that provides a use-oriented PSS. The customers use the appliances by a subscription-based mechanism without buying them. The PSS is realized by using IoT components, but the study provides only limited information about out how IoT can facilitate the usage phase and PSS maintenance.

Another group of studies has shown ways to use IoT in PSS business-model development. Zancul et al. (2016) propose a method for using IoT-enabled PSS in its two-part business model: First, they apply failure mode and effects analysis (FMEA) to decide which features of IoT should be integrated with the product. Second, they use a PSS business-strategy configurator that assists PSS providers with positioning themselves during innovation planning. They merge the results of the configurator with the FMEA approach to determine what product features and PSS processes must be implemented with the help of IoT. They apply and evaluate their method in a case study. Similarly, Shih et al. (2016) propose a PSS design method that extends visual-mapping methods for service creation incorporating IoT technology (e.g., Matzen and McAloone, 2009; Moritz, 2009). Shih et al. (2016) introduce a new concept called "pseudo-actor," which stands for an IoT-enabled object with sensors and actuators. Their method follows a six-step procedure and tackles selecting IoT technology alternatives for customer value creation. The method mostly focuses on PSS design for engineers and the study does not cover general IoT potential for PSS.

Several studies have addressed the use of IoT-for-PSS implementation. For instance, a framework for implementing industrial IoT-enabled PSS is presented by Alexopoulos et al. (2018) in support of PSS development with regard to lifecycle management and service implementation using IoT. The framework consists of various IoT-related elements selected to facilitate the service-side implementation of PSS. In a pilot case study, the framework is mapped into a real case and implementation of the IoT framework is presented. Because the focus of the framework is on IoT implementation for PSS, they do not provide an analysis of the overall capabilities of IoT-for-PSS. Similarly, Espíndola et al. (2012) address the convergence of IoT and PSS implementation by providing a conceptual design that comprises both IoT and PSS elements. In addition, they propose a middleware architecture that can realize IoT implementation with the purpose of PSS enablement. In general, the study tackles implementation details for incorporating IoT in PSS. Although this group of studies does not address overall opportunities of using IoT-for-PSS, they complement our work as they detail the implementation.

<Table 3> summarizes prior work on the relation between IoT and PSS. Existing case studies on the integration of IoT in PSS are mostly application-oriented and only partially cover the ways in which IoT supports PSS. The studies of IoT-for-PSS business-model development propose processes and methods but do not comprehensively cover all the implications of IoT-for-PSS. We found that prior

<Table 3> Summary of IoT–PSS Studies

Study Type	Study	Main Contribution
General Case Studies	Seregni et al. (2016)	Identifying IoT-enabled Services of PSS
	Elia et al. (2016)	Performance Evaluation of IoT-enabled PSS
	Bressanelli et al. (2018)	Identifying PSS challenges that IoT could overcome
IoT for PSS Business Model	Zancul et al. (2016)	A method for adopting IoT in PSS business model based on FMEA and PSS business strategy configurator
	Shih et al. (2016)	PSS design method that extends visual mapping methods for service creation by incorporating IoT
IoT for PSS Implementation	Alexopoulos et al. (2018)	IoT Framework for PSS service implementation
	Espíndola et al. (2012)	A conceptual design and a middleware architecture for incorporating IoT in PSS implementation

studies of IoT-for-PSS implementation do not consider the big picture and the use of IoT to enable PSS business models. In general, there is a lack of knowledge of what opportunities IoT can provide for PSS in general. Hence, in this study, we first build a theoretical framework that integrates different views of the opportunities for use of IoT in PSS.

Ⅲ. Study Design

To gain a deeper understanding of opportunities for use of IoT in PSS from a theoretical and a practical perspective, we conducted a structured literature review based on Vom Brocke et al. (2009) and Webster and Watson (2002) as well as expert interviews based on Gläser and Laudel (2010), Mayring (2010), and Miles and Huberman (1994). We used this mixed-method approach for completeness (Venkatesh et al., 2013), aiming to provide a comprehensive picture of the subject of interest by mixing evidence from the literature and from practice.

3.1. Systematic Literature Review

To analyze IoT opportunities for PSS from the literature, we applied the approach and instructions based on Vom Brocke et al. (2009) and Webster and Watson (2002). In this process, we used the IEEE, SpringerLink, ScienceDirect, and Scopus databases (<Table 4>). We applied the following research string: (Lifecycle OR Life-cycle OR “Life cycle”) AND (Development OR Manufacturing OR Production OR Deployment) AND (Interdisciplinary OR Multidisciplinary OR “Product Service System” OR “Cyber Physical System”) OR IoT OR “Internet of Things” OR Servitization OR Digitalization. We included all types of scientific literature without confining ourselves to a specific publication year range or ranking.

For the analysis, we first analyzed the title and abstracts and removed duplicates. We selected only relevant publications based on sets of inclusion and exclusion criteria. The exclusion criteria consisted of papers with their main focus on IoT implementation or tools. The inclusion criteria were papers addressing lifecycle management in the context of IoT and PSS and of IoT integration in business. This selection reduced the number of possibly relevant publications to 160. In the second screening, we studied the full text of the papers and evaluated their relevance to our research question. We ended up with 72 relevant papers. As a first result, we saw that only a few papers combine IoT with PSS or servitization. To cope with this issue, we interviewed experts.

3.2. Expert Interviews

As the literature review revealed some gaps, we enriched our data with expert interviews based on Gläser and Laudel (2010), Mayring (2010), and Miles and Huberman (1994). For interview sampling, we looked for leading enterprises and startups across different IoT-application fields. We chose business managers who consider or involve IoT in their processes, consultants who offer IoT solutions, and start-ups working in the IoT field. We conducted 13 semi-structured interviews (<Table 5>).

The interviews were based on a semi-structured

<Table 4> The Outcome of Database Search

Database	*Initial search*	*Title and abstract screening*	*Full-text screening*
IEEE	124	25	17
SpringerLink	1127	72	20
ScienceDirect	53	21	16
Scopus	683	42	19
Total	1987	160	72

<Table 5> Interview Details

Interview ID	Job description	Industry	Employees	Duration (min)
Participant 01	Business development manager	Global e-commerce & cloud computing	~566000	~35
Participant 02	IoT evangelist & business development manager	Global e-commerce & cloud computing	~566000	~15
Participant 03	Machine Learning Expert	Research institute	~200	~40
Participant 04	Data scientist for rail transportation	Industrial manufacturing	~372000	~10
Participant 05	Hardware product developer	Start-up in the field of automatization solutions	~12	~20
Participant 06	Innovation manager	Manufacturer of braking systems for rail and commercial vehicles	~25000	~35
Participant 07	Chief Technology Officer	Start-up in the field of digital gastronomy	~12	~45
Participant 08	Consultant for innovation & product lifecycle management	Global IT consultancy	~120000	~50
Participant 09	Product manager for digital lab and smart home	Global automotive manufacturer	~125000	~25
Participant 10	Digital E-Care	Global telecommunication company	~1800	~70
Participant 11	IoT consultant and app developer	IoT consultancy and software house	~124000	~35
Participant 12	Product manager for industrial communication	Industrial manufacturing company	~372000	~20
Participant 13	Consultant and developer	IoT consultancy	~10000	~50

interview guideline with open questions (Gläser and Laudel, 2010), to ensure some common topics and leave room for the specific aspects of every expert. Every expert was asked about general opportunities of IoT and applications in which IoT has been realized (I), opportunities and realized applications resulting from new data (II), and opportunities and realized applications for their specific processes, products or product-service systems (III). For data analysis, all interviews were transcribed and openly coded. For data analysis, all interviews were transcribed and openly coded according to Corbin et al. (2015). Our coding is shaped around two core concepts, IoT opportunities for PSS business model and IoT opportunities for PSS implementation.

The semi-structured interviews were based on guidelines (Gläser and Laudel, 2010), that both ensured some common topics and allowed for open questions, leaving room for specific aspects pertaining to each expert. Every expert was asked about general IoT opportunities and applications in which IoT has been realized, about opportunities and realized applications resulting from new data and/or pursued for their specific processes, and about products or PSS. For data analysis, all interviews were transcribed and openly coded according to Corbin et al. (2015). Our coding is focused on the two core concepts of IoT opportunities for PSS business models and for PSS implementation.

Ⅳ. Internet of Things as Product–Service System Business-Model Enabler

For the first part of the results, we present the framework of the IoT–PSS business-model opportunities (<Table 6>), which entails two dimensions: the horizontal axis stands for three general types of PSS introduced by Tukker (2004). The vertical axis presents the levels of IoT involvement in the PSS concept. The four levels are inspired by capability levels of smart products introduced by Porter and Heppelmann (2014) and cover a wide range of IoT implications from simple sensor-enabled products to complex product and service connectivity with autonomous behaviors. The transforming and optimizing levels enable IoT-driven PSS, while the interacting and tracking levels enable IoT-supported PSS. While an IoT-supported PSS is a PSS enhanced with IoT technologies, IoT fundamentally affects PSS design and implementation in an IoT-driven PSS. In

<Table 6> The Framework of IoT-PSS Business Model Opportunities

		Product-oriented PSS	Use-oriented PSS	Result-oriented PSS
IoT-Driven PSS	Transforming	Autonomous Product and Manufacturing	Continuously Improving Advanced Services	Proactive Smart Results
	Optimizing	Efficient Product and Manufacturing	Personalized Services	Smart Results
IoT-Supported PSS	Interacting	Smart Product	Engaging Services	Engaging Results
	Tracking	High Product Quality; Advanced Sales	High Service Quality; Lower Maintenance Cost	Customized results

other words, IoT is the main value creator in an IoT-driven PSS. The inner text of every cell in the framework encapsulates the potential values added by IoT for each PSS type. These values can be variously derived, as will be discussed in this section below. Also, for every concept of the framework, we provide PSS real cases.

4.1. Tracking

Tracking is the lowest level of IoT integration in PSS business models. It enables tracking of primary product, service, user and their attributes such as quality and performance metrics. The tracking capability increases awareness of not only the system but also the environment, in which the PSS is functioning (Lee et al., 2013). For instance, we can even track complex parameters such as frost risk and humidity using smart water sensors (Participant 13). Therefore, the provider would be able to add extra value by improving the quality in use for the users and decreasing the maintenance costs (Beuren et al., 2016; Zancul et al., 2016). An important implication of the tracking is reflected in product delivery phase and logistics (Barbosa et al., 2016; Papakostas et al., 2016; Porter and Heppelmann, 2014). An example of result-oriented PSS enabled by IoT is a wirelessly connected single-function button that allows customers to order products or services (Participant 02; Participant 01). Tracking and storing processes in these buttons enables us to request for the result instantly with a click of the button. Another example would be location-based services to users, which are enabled by the tracking capabilities of IoT. Th, we would be able to improve the customer experience and increase the usage or purchase rate (Participant 07).

There are plenty of PSS cases that have already realized the tracking abilities of IoT into PSS. For instance, many transportation companies that provide fleet management services have integrated IoT into their system for real-time monitoring of the vehicles and their status. These fleet management cases can be of all three types of PSS depending on their grounding business model. <Table 7> presents a real PSS case for every type of PSS. These PSS cases are built upon tracking abilities of IoT. HP Instant Ink is a result-oriented PSS that focuses on delivering the right amount of ink for HP printers. This PSS charges customers based on the number of successful prints, i.e. the desired result, instead of ink usage.

4.2. Interacting

As the next level, IoT enables a PSS to not only track and report PSS-related data, but also have some degree of action. This can be realized using an event-based scheme or direct interaction with the user. For example, in the case of a smart home PSS – in which the home devices and appliances are owned by the PSS provider and the usage is sold to the customer - the lights of a smart home can be turned on or off automatically from the outside light or the user can directly control them remotely. Similarly, the product would be able to react proactively to a particular condition. The idea is that the product has some degree of self-diagnosis and is able to interact with the user or provider. For example, the user will be instructed to replace a part in the event of an error. Such an ability increases customer engagement with the PSS (Participant 13). In general,

4) https://www.proglove.com/
5) https://www.samsara.com/uk/customers/empyre-builders
6) https://instantink.hpconnected.com
7) https://www.olimpiasplendid.com/home-automation/aquad

<Table 7> IoT Tracking-supported PSS Examples

Product-oriented PSS		Use-oriented PSS		Result-oriented PSS	
PSS Case	Description	PSS Case	Description	PSS Case	Description
ProGlove[4]	Enhancing regular gloves with barcode scanners to track information faster	Empyre Builders[5]	Using IoT-tracking to monitor construction vehicles and their movements	HP Instant Ink[6]	A system of delivering the right amount of ink whenever is needed

<Table 8> IoT Interacting-supported PSS Examples

Product-oriented PSS		Use-oriented PSS		Result-oriented PSS	
PSS Case	Description	PSS Case	Description	PSS Case	Description
AQUADUE® control[7]	System of the air-conditioning//heating installation that works smartly using IoT sensors and actuators	QIVICON[8]	Providing an IoT platform that connects various smart home devices with the monitoring and interaction features	Phillips 'Pay-per_Lux'[9]	Providing the adjusted correct amount of light (light as the result instead of selling the light bulbs)

according to the interviews, the interacting capabilities of IoT allows PSS providers to introduce new field services (Participant 04; Participant 06; Participant 07). Connected devices, simple interaction abilities with the environment and conditional clauses – provided by IoT – realize new advanced services for a PSS (Participant 10; Participant 03; Participant 04).

<Table 8> provides examples of IoT-supported PSS cases realized by interacting capabilities. As discussed, smart home solutions are common examples of this level of IoT integration that automatically react and control the situations of a house. Nevertheless, PSS providers that have incorporated interacting abilities of IoT, often do not stop at this level and utilize higher levels of IoT integration.

4.3. Optimizing

The interviewees argue that, although the tracking and interacting capabilities added by IoT support the creation of new business models, they are not sufficient (Participant 03; Participant 13). Thus, we need to involve IoT more into the development of PSS business models and the next step is optimizing capability, which is built upon the preceding capabilities. The data collected and processed during tracking and interacting allows an advanced analysis of products and services, particularly in the usage phase. This empowers PSS providers to increase the performance of products and services, decrease their costs and identify new opportunities for extending their business models (Vuppala and Kumar, 2014). Optimizing capability allows the smartness of a PSS to be dynamic and to evolve through the lifecycle (Barbosa et al., 2016). For instance, sales services become much more intelligent by analyzing the usage data in an IoT-supported PSS (Herterich et al., 2015; Zancul et al., 2016). In addition, pricing can be continuously be calculated in real-time (Zancul et al., 2016). Interviewees perceived great opportunities based on machine learning algorithms, which can improve the system functions continuously (Participant

ue-control
8) https://www.qivicon.com/
9) https://www.innovationservices.philips.com/news/philips-transition-linear-circular-economy/

<Table 9> IoT Optimizing-driven PSS Examples

Product-oriented PSS		Use-oriented PSS		Result-oriented PSS	
PSS Case	Description	PSS Case	Description	PSS Case	Description
glassbeam's Medical Device Serviceability with IoT Analytics[10]	Remote visibility and analytics leading to higher efficiency for a medical equipment provider	Rolls-Royce 'Power-by-the-H our'[11]	Engine maintenance system that uses IoT to track and analyze engines for better maintenance	ABT Power Management[12]	Adjusting and optimizing the precise amount of power provision (right amount of power is the desired result)

01; Participant 03; Participant 12). They believed such machine learning techniques combined with the connectedness of products and services over a PSS enabled by IoT provides opportunities to automate processes and create advanced solutions (Participant 07). Many interviewees emphasized the importance of optimizing with regard to control of PSS failure behavior (Participant 04; Participant 03).

Most of real PSS cases, which have integrated IoT, are more concerned about optimizing capabilities, particularly, predictive maintenance and optimized service provision. Regarding product-oriented PSS cases, availability of PSS and its maintenance services are improved significantly using analytics enabled by IoT. Moreover, result-oriented can benefit the most by optimizing the result-oriented services that they provide.

4.4. Transforming

Built on the entire IoT technology stack, the transforming capability of IoT for PSS is realized by a high level of autonomous operations and seamless communication with other networks (Gigli and Koo, 2011; Porter and Heppelmann, 2014). Transformation for the smart home example means that home appliances track their usage, perform analysis and accordingly change their behavior, interact with the user as well as other devices and the PSS provider. Therefore, there is a total connectedness and interaction among people and machines with the aim of maximizing the products performance and quality of services (Participant 09). Total IoT integration significantly reshapes the products and service provision as well as the customer's experience (Participant 01). With regard to the autonomy aspect, edge processing - processing power at the edge of the network – is a key ability. It allows local decision making for every object in the system by the collection of raw sensor data, data filtering and data processing at its source by intelligent devices (Barbosa et al., 2016; Haller et al., 2008). During the maintenance phase, the system would be able not only to warn the provider or the user but also to enable the provider to employ a predictive maintenance scheme as well as a real-time autonomous decision making (Zancul et al., 2016). To create more value, it is necessary to establish a combination of machine-learning methods with real-time and cloud-based infrastructure as well as communication across the system's network (Participant 10; Participant 03).

Complete integration of IoT into PSS allows particular intelligence for every PSS according to its history and capabilities. Such intelligence would adopt a PSS to its environmental factors, process information and usage data (Kiritsis, 2011; Porter and Heppelmann,

10) https://www.glassbeam.com/resources#casestudies
11) https://www.rolls-royce.com/media/press-releases-archive/yr-2012/121030-the-hour.aspx
12) https://www.wemanagepower.com/

<Table 10> IoT Transform-driven PSS

Product-oriented PSS		Use-oriented PSS		Result-oriented PSS	
PSS Case	Description	PSS Case	Description	PSS Case	Description
Tesla[13)]	Full autonomous driving based on connected vehicles (This is a vision and is not realized yet)	KEAZ[14)]	Using IoT to provide smart mobility and connectivity solutions	ween.ai[15)]	Providing total connectedness, autonomy, real-time predictions upon IoT devices for various solutions such as smart home or mobility services

2014) Hence, PSS providers could benefit extra value as the system autonomy and intelligence assess functionalities of the system and its components as it is running and evolving in its environment.

Few successful PSS cases could utilize the transformation ability of IoT for PSS. Many companies have envisioned such a transformation, yet it is not realized. For example, the automotive industry is working intensely on autonomous driving. The long-term vision would be the total connectedness among vehicles that smartly provide mobility services in smart cities. Although such vision has not fulfilled completely, we could not find any other real-case example for product-oriented PSS transformed by IoT. <Table 10> presents the real-case PSS cases that are driven by the transforming abilities of IoT.

Ⅴ. IoT as PSS Lifecycle Management Enabler

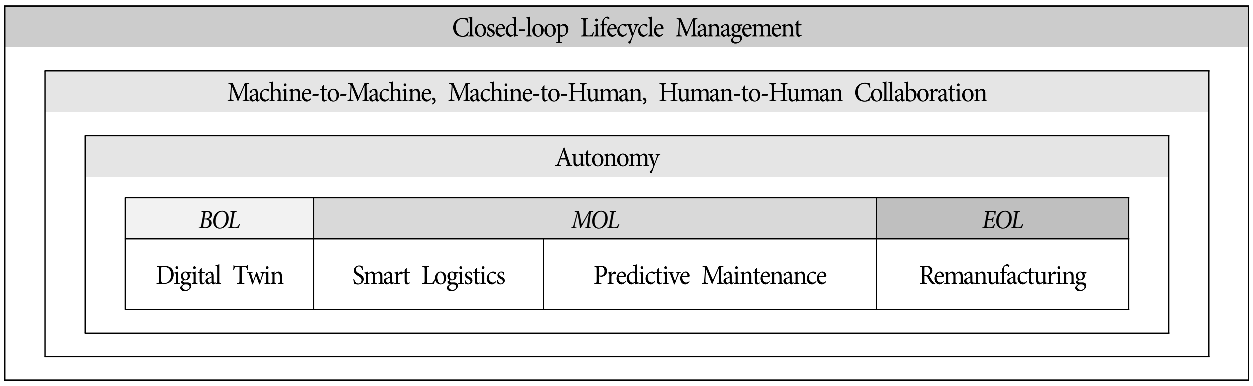
In this section, we address how IoT can facilitate PSS lifecycle management and implementation. Based on the literature and the interviews, we identified the related core potential concepts, which are presented in <Figure 1>. With regard to the overall lifecycle management, IoT provides three main capabilities. First, IoT involvement leads to an increasing amount of data belonging to the PSS and PSS development. The data can be exploited continuously for production improvement and closed-loop lifecycle management reflects this capability. The second aspect tackles collaboration issues in PSS development, which is inherently challenging due to the variety of the disciplines involved. IoT supports collaboration by enabling communication among machines and humans. Another implication of IoT for PSS development is the higher degree of autonomy for the PSS development. In addition to the overall concepts, IoT enables specific technologies and paradigms for every phase of PSS development. Regarding the PSS development phases of PSS, we follow the generally accepted distinctions between the beginning of life (BOL), middle of life (MOL) and end of life (EOL) phases. These phases represent design, manufacturing, logistics, use, maintenance, reuse and recycling, respectively (Beuren et al., 2016; Terzi et al., 2010). Throughout these phases, we identified four underlying opportunities, namely, digital twin, Closed-loop Lifecycle Management (CLLM) stands for the ubiquity of product-relevant information at any point in the lifecycle (Wiesner et al., 2015; Wuest et al., 2014). Such omnipresence enables stakeholders to track and manage the data even during the use (Kiritsis, 2011). In traditional lifecycle management, a considerable amount of relevant data is either lost or acquired at a high cost. Consequently, there is

13) https://www.tesla.com/
14) https://keaz.co/
15) https://www.ween.ai/

limited visibility of products and services for the PSS provider (Basselot et al., 2017; Igba et al., 2015). IoT tracking capabilities overcome such a challenge by low-cost collecting of relevant data among life-cycles of PSS product parts and PSS services (Basselot et al., 2017). Moreover, incorporating IoT into the PSS development would solve the challenge of low interoperability among heterogeneous working units that prevents CLLM realization (Basselot et al., 2017; Igba et al., 2015). The interviews reflected the same argument that with the help of IoT, we would collect and manage PSS-related data necessary for CLLM (Participant 01; Participant 06). PSS providers would be able to increase the quality of their product and services continuously. In addition to tracking status of a product, i.e. product-focused data, Matsas et al. (2017) introduce user-focused data, which reflect only usage-related information and attributes perceived by the user. Utilizing these two types of data can significantly support requirements elicitation and management for PSS' products and services and even introducing new ones (Gudergan et al., 2017; Wuest et al., 2016; Yang et al., 2009).

Collaboration-related aspects are challenging for PSS development as PSS development involves a high number of teams and disciplines, whose tools and methods (Gopsill et al., 2011). IoT capabilities mitigate the severity of such a challenge in collaborations among humans and machines. First, IoT-enhanced machines would be able to transfer their information and adjust their conditions to be aligned with each other. Hence, **Machine-to-Machine (M2M)** collaboration would take place without human intervention (Lee et al., 2013). With regard to **Human-to-Human (H2H)** collaboration, interviewees from a global e-commerce enterprise highlighted that employing IoT makes the relationship among manufacturers deeper as it increases the interoperability and the supply chain performance can be monitored almost in real-time (Participant 01; Participant 02). Interviewees also agreed that unleashing the potential of a complete IoT solution lead to engagement with new partners, vendors and platforms (Participant 11; Participant 03; Participant 01). In particular, tools and development platforms in the context of IoT allow a wider range of developers to access its innovative capabilities and build up their knowledge collaboratively (Participant 03). Consequently, companies can focus on their core competence and core business activities (Participant 07).

M2M collaborations enabled by IoT establish new opportunities for process and factory automation by minimizing human intervention (Ardolino et al., 2016; Gerpott and May, 2016; Lee et al., 2013).


Closed-loop Lifecycle Management
Machine-to-Machine, Machine-to-Human, Human-to-Human Collaboration
Autonomy
BOL
MOL
EOL
Digital Twin
Smart Logistics
Predictive Maintenance
Remanufacturing

<Figure 1> Opportunities of IoT for PSS Lifecycle Management

Interviews showed cases in which IoT could automate the complete supply chain processes from an order on the website to final delivery. This led to cost reduction and improved customer experience (Participant 01; Participant 02). Moreover, incorporating advanced machine learning techniques based on data collected and filtered by IoT empowers **autonomous** decision-makings, self-coordination and self-diagnosis abilities (Porter and Heppelmann, 2014), which is confirmed by the interviews (Participant 11; Participant 03). However, the interviewees argued that several challenges still impede the realization of high autonomy. For example, there are as yet no advances in automated self-criticism, in which the system recognizes its mistakes (Participant 03). In addition, there is still a lack of trust in automation operations, which prevent them from becoming fully integrated into lifecycle management (Participant 03).

Digital twin or product avatar refers to a digital equivalent of a physical product. Integrating actual physical data with the virtual replication of a product enables better design, validation and verification of engineering artifacts (Goto et al., 2016). In general, a trend can be seen toward the use of digital twin enabled with IoT capabilities (Participant 08). Digital twin can be engaged for predicting, optimizing and verifying the products along the lifecycle. However, it plays a significant role in the BOL phase by incorporating feedback from the MOL and EOL phases into improving the design and simulating different options (Participant 01; Participant 02). For instance, a digital presentation of a product supports the evaluation of product performance in diverse environments. Moreover, applying a change in PSS can first be reflected in the virtual setting and the results can be used to realize PSS more efficiently (Participant 02; Participant 08). Another important ability of digital twin is that we can present the system thoroughly and more easily to different stakeholders along the entire lifecycle (Participant 02; Participant 08). Use of digital twin reduces delays and increases both the overall development efficiency and transparency of customers' processes (Meneghetti et al., 2016).

Smart logistics is enabled by tracking and the optimizing abilities of IoT. IoT establishes an overall connectivity of all devices and product parts, which empowers the efficient delivery of products and integrated services (Vuppala and Kumar, 2014). For instance, IoT supports activities such as resource allocation (Barbosa et al., 2016) and inventory management (Papakostas et al., 2016). Moreover, with the help of IoT, autonomous vehicles would be able to optimize transportations during manufacturing and facilitate distributed orders (Mueller et al., 2017). Based on the interviews, such capabilities of IoT are currently in use in several manufacturing leaders (Participant 01).

Predictive maintenance is regular monitoring and analyzing of the system conditions in order to minimize the number of failures and repairs (Mobley, 2002). Since IoT provides valuable insight with regard to the PSS and its usage, it can minimize the time for error diagnosis (Lerch and Gotsch, 2015). For example, with the help of IoT sensors and analysis of the collected usage data, we would be able to elicit spare part requirements (Herterich et al., 2015; Zancul et al., 2016) Several interviewees reported that they have experienced considerable savings by incorporating IoT capabilities in their maintenance activities (Participant 01; Participant 08; Participant 05). Moreover, they stated that increased availability resulting from more efficient maintenance led to higher customer satisfaction.

Remanufacturing stands for the industrial process, in which we restore and recover used products into a good condition (Lindkvist and Sundin, 2016). With this, the experiences from the later stages of a lifecycle would be employed in the earlier stages (Igba et al., 2015). Realizing remanufacturing necessitates tracking, controlling and analysis of the product, its condition and its usage, which can be enabled by means of IoT (Chierici and Copani, 2016). Ideally, a feedback loop would be in place between each lifecycle phases.

Ⅵ. Discussion

The IoT paradigm has the potential to transform the industry and be as influential as the Internet was in the 1990s. Our findings showed that practitioners assert the high potential of IoT for facilitating new business models, designing new products and providing advanced services. In conformance with this fact, the prior research emphasized on transforming abilities of IoT and the big impact that it can have on businesses (Čolaković and Hadžialić, 2018; Gubbi et al., 2013; Porter and Heppelmann, 2014). In particular, IoT can play a crucial role in PSS development (Seregni et al., 2016; Shih et al., 2016; Zancul et al., 2016). Due to challenging nature of PSS, which transforms merely product or service businesses into an integrated enterprise of product and service provision, more connectedness and communication among heterogeneous elements is necessary (Vasantha et al., 2012; Wiesner et al., 2015). The strengths of IoT match the difficulties that PSS design and development confront.

The existing studies of the IoT and PSS relationship have been limited to single case applications of a particular method for adopting IoT in PSS development (Shih et al., 2016; Zancul et al., 2016). We extend the current literature by establishing a comprehensive view of the opportunities that IoT can provide for PSS. We have presented the framework of IoT-PSS business model opportunities that introduces four levels of IoT involvement in PSS. Based on the framework, there is a wide range of IoT integration into PSS. It starts from basic IoT-supported tracking abilities in PSS and proceeds to the most-complex abilities, the transformed IoT-driven PSS with IoT as its core value creator. The framework assists PSS providers in positioning themselves, identifying the extent, to which they have already benefited from IoT and the possibilities that they have not yet realized. Furthermore, we identified and highlighted the core IoT-enabled opportunities, which facilitate PSS lifecycle management. Although the concepts vary largely from M2M collaboration to digital twin and remanufacturing, they are mutual in terms of being enabled by IoT and advancing PSS lifecycle management. Nevertheless, diving deep into the details of implementing such technologies in the domain of PSS was beyond the scope of this study but can be investigated in future research. We argue that our study provides the fundamentals for advancing PSS and IoT integration research. Future studies can build new concepts, methods, and tools upon the frameworks established in this study.

Combining the two folds of this study's contribution enlighten the overall IoT exploitation for PSS design and development. The insightful alignment of IoT and PSS brings various added-values for both businesses and customers. Regarding the customer values, PSS providers would be able to establish a reliable connection with the customer, partners and suppliers by a right IoT integration. Customers can expect continually improving products and services, which are also more customized to their usage. In addition, customers would benefit from higher availability of product and services. In the context of the business values, IoT integration shortens the development cycles and reduces costs of development. PSS providers will have a shorter time-to-market, which is a decisive aspect in a competitive environment. Moreover, utilizing IoT decreases the costs of maintenance and remanufacturing significantly. For example, there would be no need for on-site monitoring of product conditions as the sensors are continuously tracking the relevant information. At its extreme realization, PSS providers will gain autonomy and transparency during all phases of PSS lifecycle. Even though a limited integration of IoT in PSS enables PSS providers to introduce smart products and advanced services, allowing them to increase revenue.

Furthermore, we could identify major challenges in the use of IoT for PSS that future studies should tackle them. First, although IoT can facilitate collaboration among humans and machines, it may also add extra complexity to PSS development as IoT implementation necessitates integration and collaboration of various knowledge experts (Participant 12; Participant 04). Moreover, IoT implementation may shape new partnerships due to its technical complexity. This brings new inter-organization collaborations for PSS providers. Another challenge is finding the right methodology to develop IoT-supported and IoT-driven PSS. For example, alignment between the simultaneous development of software and hardware have difficulties for enterprises (Participant 02; Participant 09). Hence, future research should establish new methods that can tackle such challenges. Finally, huge captured, generated and collected data in IoT-driven PSS have to be managed consistently. Establishing interoperability among various tools, artifacts and data sources is a difficult goal to achieve. Therefore, future research needs to investigate interoperability in IoT-driven PSS and mechanisms to achieve it.

According to our findings from the interviews, IoT technologies have been integrated mostly on the end-customers side, even though B2B applications of IoT can have greater economic outcomes. Moreover, we observed slow progress regarding the shift from IoT-supported PSS to IoT-driven PSS. The lack of infrastructural capabilities can be considered an important factor hindering IoT integration, but the future studies should investigate in detail the existing complicated barriers exist. For example, there is still uncertainty about costs and profits associated with IoT adoption, particularly at its highest extent. Mechanisms to analyze and estimate IoT adoption in terms of monetary parameters would significantly support the realization of IoT opportunities. Furthermore, IoT integration is fostering a collaborative ecosystem, in which many start-ups have emerged as IoT technology providers. Future studies can look more into how we can ease the integration of such start-ups' contributions into existing infrastructures. With this regard, the research should study the role of emerging IoT platforms, which will facilitate the use of IoT for a variety of applications.

A limitation of our work was using interview and literature review in a mixed-method approach, while a mixed-method approach is most fruitful when qualitative and quantitative methods are combined. Therefore, we suggest future empirical studies to employ quantitative and qualitative methods for addressing IoT for PSS topic. To this end, researchers can conduct a quantitative analysis of successful cases of IoT and PSS integration which would be complemented by further in-detail case studies.

Ⅶ. Conclusion

IoT technologies are changing products, services and the way we develop them. In addition to empowering the existing solutions, IoT enables us to realize new ideas. Particularly, we can use the power of IoT to facilitate complexity of PSS design and development. In this study, we investigated opportunities that IoT can provide for PSS business models and lifecycle management. We provided examples of each relevant hotspot to assist PSS providers in positioning and deciding the right business model when integrating IoT in their portfolio. First, we introduced the framework of IoT opportunities for PSS business models that entails two dimensions of IoT involvement level and PSS types. It evaluates which type of services IoT technologies foster for the provision of PSS. Furthermore, we analyzed IoT as a key facilitator of the lifecycle management by enabling new technologies and capabilities such as autonomy, closed-lifecycle management, digital twin, predictive maintenance and remanufacturing.

The findings of this study provide new insights for PSS providers. The study increases their awareness regarding the potentials of IoT for PSS and their current progress of IoT realization. Moreover, this study establishes a comprehensive view on opportunistic implications of IoT for PSS, which paves the way for future studies to advance this topic. The research can complete this work by addressing, on one hand, the barriers for integrating IoT into PSS and on the other hand, the challenges caused by IoT integration into PSS. Accordingly, the studies can propose solutions to overcome such challenges.

Acknowledgement

This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) collaborative research centre 'Sonderforschungsbereich SFB 768 "Managing cycles in innovation processes - Integrated development of product-service-systems based on technical products".

<References>

[1] Adrodegari, F., and Saccani, N. (2017). Business models for the service transformation of industrial firms. *The Service Industries Journal, 37*(1), 57-83.

[2] Alexopoulos, K., Koukas, S., Boli, N., and Mourtzis, D. (2018). Architecture and development of an Industrial Internet of Things framework for realizing services in Industrial Product Service Systems. *Procedia CIRP, 72*, 880-885. doi:https://doi.org/10.1016/j.procir.2018.03.152

[3] Ardolino, M., Saccani, N., Gaiardelli, P., and Rapaccini, M. (2016). Exploring the key enabling role of digital technologies for PSS offerings. *Procedia CIRP, 47*, 561-566.

[4] Baines, T. S., Lightfoot, H. W., Evans, S., Neely, A., Greenough, R., Peppard, J., ... Tiwari, A. (2007). State-of-the-art in product-service systems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 221*(10), 1543-1552.

[5] Bandyopadhyay, D., and Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications, 58*(1), 49-69.

[6] Barbosa, J., Leitão, P., Trentesaux, D., Colombo, A. W., and Karnouskos, S. (2016). Cross benefits from cyber-physical systems and intelligent products for future smart industries. Paper presented at *the Industrial Informatics (INDIN), 2016 IEEE 14th*

International Conference on.

[7] Basirati, M. R., Weking, J., Hermes, S., Böhm, M., and Krcmar, H. (2019). IoT as PSS Enabler: Exploring Opportunities for Conceptualization and Implementation. Paper presented at *the PACIS, Xi'an*, China.

[8] Basirati, M. R., Zou, M., Bauer, H., Kattner, N., Reinhart, G., Lindemann, U., ... Vogel-Heuser, B. (2018). Towards systematic inconsistency identification for product service systems. Paper presented at *the Proceedings of the DESIGN 2018 15th International Design Conference.*

[9] Basselot, V., Berger, T., and Sallez, Y. (2017). Active Monitoring of a Product: A Way to Solve the "Lack of Information" Issue in the Use Phase. In *Service Orientation in Holonic and Multi-Agent Manufacturing* (pp. 337-346), Springer.

[10] Beuren, F. H., Ferreira, M. G. G., and Miguel, P. A. C. (2013). Product-service systems: a literature review on integrated products and services. *Journal of Cleaner Production, 47*, 222-231.

[11] Beuren, F. H., Pereira, D., and Fagundes, A. B. (2016). Product-service systems characterization based on life cycle: application in a real situation. *Procedia CIRP, 47*, 418-423.

[12] Bressanelli, G., Adrodegari, F., Perona, M., and Saccani, N. (2018). The role of digital technologies to overcome Circular Economy challenges in PSS Business Models: an exploratory case study. *Procedia CIRP, 73*, 216-221. doi:https://doi.org/10.1016/j.procir.2018.03.322

[13] Chierici, E., and Copani, G. (2016). Remanufacturing with Upgrade PSS for New Sustainable Business Models. *Procedia CIRP, 47*, 531-536.

[14] Čolaković, A., and Hadžialić, M. (2018). Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Computer Networks.*

[15] Corbin, J., Strauss, A., and Strauss, A. L. (2015). *Basics of qualitative research.* sage.

[16] Elia, V., Gnoni, M. G., and Tornese, F. (2016). Assessing the efficiency of a PSS solution for waste collection: a simulation based approach. *Procedia CIRP, 47*, 252-257.

[17] Espíndola, D., Duarte, N., Botelho, S., Carvallho, J., and Pereira, C. (2012). Internet of things to provide scalability in product-service systems. Paper presented at *the Proceedings of UBICOMM 2012: The Sixth International Conference on Mobile Ubiquitous Computing*, Systems, Services and Technologies.

[18] Exner, K., Zimpfer, R., and Stark, R. (2017). Maturity model and action recommendation: a PSS capability self-assessment tool for companies. *Procedia CIRP, 64*, 175-180.

[19] Georgakopoulos, D., and Jayaraman, P. P. (2016). Internet of things: from internet scale sensing to smart services. *Computing, 98*(10), 1041-1058.

[20] Gerpott, T. J., and May, S. (2016). Integration of Internet of Things components into a firm's offering portfolio – a business development framework. *Info, 18*(2), 53-63.

[21] Gigli, M., and Koo, S. G. (2011). Internet of Things: Services and Applications Categorization. *Adv. Internet of Things, 1*(2), 27-31.

[22] Gläser, J., and Laudel, G. (2010). *Experteninterviews und qualitative Inhaltsanalyse.* Springer-Verlag.

[23] Goedkoop, M. J., Van Halen, C. J., Te Riele, H. R., and Rommens, P. J. (1999). Product service systems, ecological and economic basics. *Report for Dutch Ministries of environment (VROM) and economic affairs (EZ), 36*(1), 1-122.

[24] Gopsill, J. A., McAlpine, H. C., and Hicks, B. J. (2011). Learning from the lifecycle: The capabilities and limitations of current product lifecycle practice and systems. Paper presented at *the DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design*, Vol. 6: Design Information and Knowledge, Lyngby/Copenhagen, Denmark.

[25] Goto, S., Yoshie, O., and Fujimura, S. (2016). Internet of Things value for mechanical engineers and evolving commercial product lifecycle management system. Paper presented at *the Industrial Engineering and Engineering Management (IEEM), 2016 IEEE International Conference* on.

[26] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*(7), 1645-1660.

[27] Gudergan, G., Buschmeyer, A., Feige, B. A., Krechting, D., Bradenbrink, S., and Mutschler, R. (2017). Value of Lifecycle Information to Transform the Manufacturing Industry. In *Shaping the Digital Enterprise* (pp. 173-194), Springer.

[28] Guth, J., Breitenbücher, U., Falkenthal, M., Fremantle, P., Kopp, O., Leymann, F., and Reinfurt, L. (2018). A detailed analysis of IoT platform architectures: concepts, similarities, and differences. In *Internet of Everything* (pp. 81-101), Springer.

[29] Haller, S., Karnouskos, S., and Schroth, C. (2008). The internet of things in an enterprise context. Paper presented at *the Future Internet Symposium.*

[30] Herterich, M. M., Uebernickel, F., and Brenner, W. (2015). The impact of cyber-physical systems on industrial services in manufacturing. *Procedia CIRP, 30*, 323-328.

[31] Igba, J., Alemzadeh, K., Gibbons, P. M., and Henningsen, K. (2015). A framework for optimising product performance through feedback and reuse of in-service experience. *Robotics and Computer-Integrated Manufacturing, 36*, 2-12.

[32] Kiel, D., Arnold, C., and Voigt, K.-I. (2017). The influence of the Industrial Internet of Things on business models of established manufacturing companies – A business level perspective. *Technovation, 68*, 4-19.

[33] Kiritsis, D. (2011). Closed-loop PLM for intelligent products in the era of the Internet of things. *Computer-Aided Design, 43*(5), 479-501.

[34] Kowalkowski, C., Windahl, C., Kindström, D., and Gebauer, H. (2015). What service transition? Rethinking established assumptions about manufacturers' service-led growth strategies. *Industrial Marketing Management, 45*, 59-69.

[35] Lee, G. M., Crespi, N., Choi, J. K., and Boussard, M. (2013). Internet of things. In *Evolution of Telecommunication Services* (pp. 257-282), Springer.

[36] Lerch, C., and Gotsch, M. (2015). Digitalized product-service systems in manufacturing firms: A case study analysis. *Research-Technology Management, 58*(5), 45-52.

[37] Lightfoot, H., Baines, T., and Smart, P. (2013). The servitization of manufacturing: A systematic literature review of interdependent trends. *International Journal of Operations & Production Management, 33*(11/12), 1408-1434.

[38] Lindkvist, L., and Sundin, E. (2016). The role of Product-Service Systems regarding information feedback transfer in the product life-cycle including remanufacturing. *Procedia CIRP, 47*, 311-316.

[39] Maleki, E., Belkadi, F., Zhang, Y., and Bernard, A. (2017). Towards a new collaborative framework supporting the design process of industrial Product Service Systems. In *Advances on Mechanics, Design Engineering and Manufacturing* (pp. 139-146), Springer.

[40] Marilungo, E., Papetti, A., Germani, M., and Peruzzini, M. (2017). From PSS to CPS design: a real industrial use case toward industry 4.0. *Procedia CIRP, 64*, 357-362.

[41] Marques, G., Garcia, N., and Pombo, N. (2017). A survey on IoT: architectures, elements, applications, QoS, platforms and security concepts. In *Advances in Mobile Cloud Computing and Big Data in the 5G Era* (pp. 115-130), Springer.

[42] Matsas, M., Pintzos, G., Kapnia, A., and Mourtzis, D. (2017). An integrated collaborative platform for managing product-service across their life cycle. *Procedia CIRP, 59*, 220-226.

[43] Matzen, D., and McAloone, T. C. (2009). *A systematic apporach to service oriented product development.* DTU Management.

[44] Maxwell, D., Sheate, W., and Van Der Vorst, R. (2006). Functional and systems aspects of the sustainable product and service development approach for industry. *Journal of Cleaner Production, 14*(17), 1466-1479.

[45] Mayring, P. (2010). Qualitative inhaltsanalyse. In *Handbuch qualitative Forschung in der Psychologie* (pp. 601-613), Springer.

[46] Mazhelis, O., Luoma, E., and Warma, H. (2012). Defining an internet-of-things ecosystem. In *Internet of Things, Smart Spaces, and Next Generation Networking* (pp. 1-14), Springer.

[47] Meier, H., Roy, R., and Seliger, G. (2010). Industrial product-service systems—IPS2. *CIRP Annals-Manufacturing Technology, 59*(2), 607-627.

[48] Meneghetti, A., Moro, S., and Helo, P. (2016). Intermixed product and service boundaries: exploring servitization in sheet metal industry. *Procedia CIRP, 47*, 258-263.

[49] Miles, M. B., and Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook.* sage.

[50] Mobley, R. K. (2002). *An introduction to predictive maintenance.* Elsevier.

[51] Mont, O. K. (2002). Clarifying the concept of product–service system. *Journal of Cleaner Production, 10*(3), 237-245.

[52] Moritz, S. (2009). *Service design: Practical access to an evolving field.* Lulu.com.

[53] Mueller, E., Chen, X.-L., and Riedel, R. (2017). Challenges and Requirements for the Application of Industry 4.0: A Special Insight with the Usage of Cyber-Physical System. *Chinese Journal of Mechanical Engineering, 30*(5), 1050-1057.

[54] Papakostas, N., O'Connor, J., and Byrne, G. (2016). Internet of things technologies in manufacturing: Application areas, challenges and outlook. Paper presented at *the Information Society (i-Society), 2016 International Conference* on.

[55] Patel, K. K., and Patel, S. M. (2016). Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. *International Journal of Engineering Science and Computing, 6*(5).

[56] Porter, M. E., and Heppelmann, J. E. (2014). How smart, connected products are transforming competition. *Harvard Business Review, 92*(11), 64-88.

[57] Reim, W., Parida, V., and Örtqvist, D. (2015). Product -Service Systems (PSS) business models and tactics - A systematic literature review. *Journal of Cleaner Production, 97*, 61-75. doi:10.1016/j.jclepro.2014.07.003

[58] Schuh, G., Salmen, M., Kuhlmann, T., and Wiese, J. (2016). Life-Cycle-Oriented Product-Service-Systems in the Tool and Die Making Industry. *Procedia CIRP, 47*, 555-560.

[59] Seregni, M., Sassanelli, C., Cerri, D., Zanetti, C., and Terzi, S. (2016). The impact of IoT technologies on product-oriented PSS: The "home delivery" service case. Paper presented at *the Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum* on.

[60] Shih, L.-H., Lee, Y.-T., and Huarng, F. (2016). Creating customer value for product service systems by incorporating internet of things technology. *Sustainability, 8*(12), 1217.

[61] Tao, F., Zuo, Y., Da Xu, L., and Zhang, L. (2014). IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Transactions on Industrial Informatics, 10*(2), 1547-1557.

[62] Terzi, S., Bouras, A., Dutta, D., Garetti, M., and Kiritsis, D. (2010). Product lifecycle management-from its history to its new role. *International Journal of Product Lifecycle Management, 4*(4), 360-389.

[63] Tukker, A. (2004). Eight types of product-service system: Eight ways to sustainability? Experiences from SusProNet. *Business Strategy and the Environment, 13*(4), 246-260.

[64] Ulaga, W., and Reinartz, W. J. (2011). Hybrid offerings: how manufacturing firms combine goods and services successfully. *Journal of Marketing, 75*(6), 5-23.

[65] Vasantha, G. V. A., Roy, R., Lelah, A., and Brissaud, D. (2012). A review of product-service systems design methodologies. *Journal of Engineering Design, 23*(9), 635-659.

[66] Venkatesh, V., Brown, S. A., and Bala, H. (2013). Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in information systems. *MIS Quarterly*, 21-54.

[67] Vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., and Cleven, A. (2009). Reconstructing the giant: On the importance of rigour in documenting the literature search process. Paper presented at *the ECIS*.

[68] Vuppala, S. K., and Kumar, H. K. (2014). Service Applications-Exploiting the Internet of Things. Paper presented at *the Global Conference (SRII), 2014 Annual SRII*.

[69] Webster, J., and Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, xiii-xxiii.

[70] Weking, J., Brosig, C., Böhm, M., Hein, A., and Krcmar, H. (2018). Business Model Innovation Strategies for Product Service Systems - An Explorative Study in the Manufacturing Industry. Paper presented at *the Twenty-Sixth European Conference on Information Systems* (ECIS 2018).

[71] Whitmore, A., Agarwal, A., and Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. *Information Systems Frontiers, 17*(2), 261-274.

[72] Wiesner, S., Freitag, M., Westphal, I., and Thoben, K.-D. (2015). Interactions between service and product lifecycle management. *Procedia CIRP, 30*, 36-41.

[73] Wortmann, F., and Flüchter, K. (2015). Internet of things. *Business & Information Systems Engineering, 57*(3), 221-224.

[74] Wuest, T., and Wellsandt, S. (2016). Design and Development of Product Service Systems (PSS)-Impact on Product Lifecycle Perspective. *Procedia Technology, 26*, 152-161.

[75] Wuest, T., Hribernik, K., and Thoben, K.-D. (2014). Capturing, managing and sharing product information along the lifecycle for design improvement. Paper presented at the *Proceedings of the 10th International Workshop on Integrated Design Engineering*.

[76] Yang, L., Xing, K., and Lee, S. (2010, 15-17 July 2010). A new conceptual life cycle model for Result-Oriented Product-Service System development. Paper presented at the *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics*.

[77] Yang, X., Moore, P., Pu, J.-S., and Wong, C.-B. (2009). A practical methodology for realizing product service systems for consumer products. *Computers & Industrial Engineering, 56*(1), 224-235.

[78] Zancul, E. d. S., Takey, S. M., Barquet, A. P. B., Kuwabara, L. H., Cauchick Miguel, P. A., and Rozenfeld, H. (2016). Business process support for IoT based product-service systems (PSS). *Business Process Management Journal, 22*(2), 305-323.

◆ About the Authors ◆

Mohammad R. Basirati

Mohammad R. Basirati (mohammadreza.basirati@tum.de) is a research associate and Ph.D. student at the chair for information systems at the Technical University of Munich (TUM). He holds a master's degree from TUM in informatics and a bachelor's degree from University of Tehran in computer engineering. His research focus is on system engineering, in particular requirements management with emphasis on collaboration aspects among stakeholders and resolution of emergent inconsistencies.

Jörg Weking

Jörg Weking (joerg.weking@tum.de) is a research associate and Ph.D. student at the Chair for Information Systems, Technical University of Munich (TUM), Germany. He studied Information Systems at the University of Münster, Germany, at the Turku School of Economics, Turku, Finland (TSE) and the Queensland University of Technology, Brisbane, Australia (QUT). His research focuses on business model patterns, business model innovation, value co-creation and product service systems. His work has been published in Electronic Markets, Communications of the AIS and in refereed conference proceedings such as the European Conference on Information Systems and the Americas Conference on Information Systems.

Sebastian Hermes

Sebastian Hermes (sebastian.hermes@tum.de) is a research associate and Ph.D. student at the Chair for Information Systems, Technische Universität München (TUM), Munich, Germany. He holds a Master's degree in Entrepreneurship from the University of Liechtenstein and a Bachelor's degree from the Baden-Wuerttemberg Cooperative State University. Sebastian has worked five years at Roche and co-founded Thinkfield. His work has appeared in conference proceedings such as Hawaii International Conference on System Sciences.

Markus Böhm

Markus Böhm (markus.boehm@tum.de) is research group leader at the Chair for Information Systems at Technical University of Munich (TUM), Germany. He graduated in Business & Information Systems Engineering from Friedrich-Alexander University Erlangen-Nürnberg (FAU), and holds a doctoral degree in Information Systems from TUM. His research focus is on mergers & acquisitions, business model innovation and digital transformation. His work has appeared in Management Information Systems Quarterly Executive, Electronic Markets, Communications of the AIS and several refereed conference proceedings, including ECIS and ICIS.

Helmut Krcmar

Helmut Krcmar (helmut. krcmar@tum.de) is a Chair Professor of Information Systems at Technical University of Munich (TUM), Germany. Before 2002, he was Chair for Information Systems, University of Hohenheim, Stuttgart. Helmut is an AIS Fellow and has served the IS community in many roles, including as President of the Association for Information Systems. His research interests include information and knowledge management, service management, business process management, and business information systems. His work has appeared in Management Information Systems Quarterly, Journal of Management Information Systems, Journal of Strategic Information Systems, Journal of Management Accounting Research, Journal of Information Technology, Information Systems Journal, and Business & Information Systems Engineering.

Submitted: March 19, 2019; 1st Revision: July 28, 2019; 2st Revision: August 26, 2019; Accepted: August 29, 2019

Not Included in
Review and
Evaluation: P5

The Journal of Systems & Software 144 (2018) 342–355

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Introducing TRAILS: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development

Thomas Wolfenstetter[*], Mohammad R. Basirati, Markus Böhm, Helmut Krcmar

Chair for Information Systems, Technische Universität München, Germany

ARTICLE INFO

Keywords:
Model-based systems engineering
Traceability
Product service systems
Model integration

ABSTRACT

Developing state of the art product service systems (PSS) requires the intense collaboration of different engineering domains, such as mechanical, software and service engineering. This can be a challenging task, since each engineering domain uses their own specification artefacts, software tools and data formats. However, to be able to seamlessly integrate the various components that constitute a PSS and also being able to provide comprehensive traceability throughout the entire solution life cycle it is essential to have a common representation of engineering data.

To address this issue, we present TRAILS, a novel software tool that joins the heterogeneous artefacts, such as process models, requirements specifications or diagrams of the systems structure. For this purpose, our tool uses a semantic model integration ontology onto which various source formats can be mapped. Overall, our tool provides a wide range of features that supports engineers in ensuring traceability, avoiding system inconsistencies and putting collaborative engineering into practice. Subsequently, we show the practical implementation of our approach using the case study of a bike sharing system and discuss limitations as well as possibilities for future enhancement of TRAILS.

1. Introduction

1.1. Motivation

In an increasingly digitised economy more and more companies realize that products themselves are no more the main contributors to value creation in their business. Instead, value for the customer is being created in service-oriented business models. Already today, most developed economies owe a far greater share of their national income to services than to manufacturing of physical products (Meier et al., 2010). Even in traditional manufacturing industries, global competition forces companies to focus on building long-term relationships with their customers by providing product-supporting services, such as maintenance, or offering the product itself as a service (Marques et al., 2013). Furthermore, environmental considerations cause enterprises to move from a product-based economy to a service-based economy which limits their susceptibility to environment issues (Maussang et al., 2009). As a consequence, the concept of product service systems (PSS), i.e. integrated systems that combine product and service components, is gaining popularity as a strategic measure to deal with these issues.

PSS development thus involves various stakeholders from different engineering domains who need to develop hardware, software and service components based on descriptions of the customers' needs and seamlessly integrate them into a comprehensive solution while at the same time reacting flexibly to changing requirements and a dynamic system environment. For example, changing legislation regarding privacy protection might impact the way customer related data is handled by the PSS provider in order to ensure compliance. This not only impacts the service processes in which this data is being collected, but also software systems that store and process the data and even might force the PSS provider to change hardware components that rely on customer data in order to provide their functions. In this example a simple requirements change entails an adaptation or possibly redevelopment of various components of the PSS, requiring engineers from different disciplines to communicate with each other, coordinate the changes made to the system as a whole and anticipate how changing one component influences other parts of the PSS. As a result, not only the degree of involvement of stakeholders from different domains increases. There is also need for tight collaboration and communication among all stakeholders involved. Therefore, a major challenge for PSS

[*] Corresponding author.
E-mail address: twolfenstetter@gmail.com (T. Wolfenstetter).

https://doi.org/10.1016/j.jss.2018.06.079
Received 1 June 2017; Received in revised form 28 April 2018; Accepted 30 June 2018
Available online 03 July 2018

engineering is to provide integrated conceptual models and comprehensive representation techniques to support cross-domain collaboration (Vasantha et al., 2012).

Moreover, the cross-domain engineering process is not the only aspect that differentiates the development of PSS from traditional product development. By its very idea the concept of PSS entails the integration of business models, products and services along the entire life cycle to create additional value for the customer (Vasantha et al., 2012). Like in every long-term relationship, expectations and capabilities, both on the provider and on the customer side evolve over time. Consequentially, PSS providers need to deal with changing requirements to be satisfied. Therefore, they need to monitor the traceability relationships between requirements and affected parts of the PSS solution including both, tangible product components as well as intangible services (Maussang et al., 2009).

The complexity of PSS engineering also manifests itself in the heterogeneity of artefacts, which are created and used along the PSS life cycle. For instance, in the process of developing a PSS every engineering domain involved follows their own domain-specific approaches when creating the various types of development artefacts that are required along the process, such as process models, requirements specifications, design structure matrices, use case diagrams or component diagrams. As artefact we hereby understand every tangible information object that is created along the life cycle of a system to describe its e.g. design, architecture, functions as well as the processes and the organisation associated with it. All of these artefacts are highly interdependent as they ultimately specify components of the PSS, which finally need to function together reliably.

Managing the relationships between PSS engineering artefacts is necessary for developers to anticipate the change impact of an artefact on others and to prevent inconsistencies as well as to trace the evolution of the individual artefacts and the PSS as a whole. For this purpose, the structural architecture of a PSS together with the dynamics of the service processes and the evolution of requirements that are linked to them needs to be captured. Moreover, all of this engineering knowledge needs to be presented in a way that allows engineers to get a comprehensive overview of the problem as well as the solution domain (Meier et al., 2010). By doing so, it is possible to dynamically adapt the solution to changing customers' needs and the evolving environment in which the PSS competes.

However, current industry practice shows that PSS engineering relies on a multitude of different modelling languages and tools that are largely incompatible with each other. Thus, today the analysis of dependencies within a PSS requires tremendous manual efforts and the integrability of components and their mutual impact can only be checked late in the development process.

In a nutshell, PSS development is a complex process with high number of dependencies between heterogeneous artefacts. In practice, traditional engineering methods often struggle when coping with the challenges of PSS development, thus producing callow solution designs that cannot live up to their full potential. Although there exists a wide range of approaches for modelling the different components of a PSS (Vasantha et al., 2012; Meier et al., 2010), the design of integrated products and services along with the issue of traceability has not been supported sufficiently by software tools (Baines et al., 2007; Cavalieri and Pezzotta, 2012; Meier et al., 2010). Also, tools and modelling languages which are used in PSS engineering do not support integrated analysis of PSS artefacts and their relationships which can lead to inconsistencies or unanticipated changes even in late phases of development. Thus we conclude that there is a lack of tool support regarding modelling and analysing PSS artefacts and their relationships from a holistic viewpoint.

1.2. Approach

We tackle this issue by proposing a tool that supports PSS development by providing means to integrate the various domain-specific artefacts into a comprehensive ”semantic engineering” graph. This graph represents the various PSS artefacts, such as requirements, components, processes, activities, stakeholders or tests as nodes and the relationships and flows between those artefacts as edges. The tool facilitates capturing the relations between different artefacts along the entire PSS life cycle. It further visualises the semantic engineering graph or particular views (subset of nodes or edges) to the user and provides features to analyse and edit the graph.

To achieve the aforementioned goal, our research intends to establish a theoretical foundation and then presents the corresponding software tool that enables cross-domain traceability and model integration among PSS elements. Based on this agenda, we name our software tool TRAILS, **Tra**ceability, model **I**ntegration and **L**ife-cycle management **S**upport.

Our proposed approach is not focused on capturing and describing every little detail of the system components that can be modelled in the respective domain-specific modelling languages (e.g. single function calls in software code, detailed geometry of hardware parts or activities of a service process that are modelled exact to the second), but it is more concentrated on a project management level, allowing requirements engineers or project managers to analyse the overall relationships between system components, requirements, stakeholders or other artefacts that are relevant in the development process. At this point, we also want to emphasize, that our approach and tool are primarily designed to support the model-based engineering (MBE) of PSS but not model-driven engineering (MDE), i.e. automatic generation of software code or service guidelines from models. In case of PSS we think that at the moment MBE is more feasible then MDE since PSS are complex socio-technical systems. Therefore models can play an important but not a dominant role in the design and development of PSS. Since PSS development requires an integrated view on the system under development, the tool features multiple integration approaches demonstrating the result as a semantic engineering graph (network). This approach is enhanced with allowing multiple views on the resulted graph for each specific purpose.

To this end, first, we define a reference ontology that specifies the conceptual entities that are used in the multiple engineering domains involved in PSS engineering. The development of this integration ontology, is based on literature reviews within the engineering domains involved as well as expert interviews, PSS case studies and modelling workshops. In TRAILS, this integration ontology is used as a framework that defines element types and their associations that are used during PSS engineering. In TRAILS, model integration is performed by transforming each instances of models whose type is supported by TRAILS into the format defined by the integration ontology and then linking similar or related artefacts.

1.3. Structure of article

The structure of this paper is as follows. In Section 2, first we give an overview of available modelling methods used in PSS engineering, then existing software tools for PSS are analysed. Afterwards, in Section 3, the model transformation methods are discussed and we explain the model transformation process used in TRAILS. Following, in Section 4, the basic concepts of TRAILS are introduced which is followed by describing available features of our tool in Section 5. In Section 6, a case study demonstrates the use of TRAILS' different features. In Section 7 we discuss limitations and possible future improvements of our approach. Finally, Section 8 gives a summary of the research presented in this paper.

2. Related work

Analysing the literature related to PSS design and development we find that research is more focused on methodologies and modelling techniques rather than providing tools to support the presented methods. Except two works on computer aided design tools targeted at PSS development as well as one publication introducing a knowledge management tool for PSS engineering, we did not find any additional tools that are explicitly designed to support the development of PSS. In this section we first summarise proposed PSS modelling methods in literature and then briefly explain the tools we found.

2.1. PSS modelling methods

A considerable number of modelling methods for PSS development have been proposed in literature, most of which aim at systematising the functions or value proposition of a PSS from the customer's perspective and focusing on the service aspect of the PSS (Qu et al., 2016). Here, one group of methods focuses particularly on the hierarchical configuration of a PSS from services or other components (Klingner and Becker, 2015). In contrast, some works aimed at covering PSS innovation phases comprehensively, therefore they offered several modelling techniques, each focusing at a particular situation in a PSS development.

Most of the approaches for modelling a PSS presented in literature are based on service blueprinting proposed by Shostack (Lynn Shostack, 1982) more than 30 years ago. Geum and Park (Geum and Park, 2011) for example extend the service blueprint with new notations to capture the flow of product usage and service usage from the provider to the customer and the relationship between products and services. Lee and Kim (2010) focus on functional modelling of a PSS. They modify the service blueprint by adding a function layer to show interactions between service provider and service consumer more explicitly. Geng and Chu (2011) use a conceptual service blueprint adding a user task model (to improve process-oriented design of a PSS with requirement analysis from user perspective) and a function model (to show the relation between requirements and PSS concepts). Service blueprinting and its extensions are mainly focused on visualisation of service processes in the context of a PSS. This technique elaborates the provided service activities at every stage of the PSS life cycle and specifies the level of customer involvement or visibility of certain activities to the customer. Although adopting service blueprint gives a thorough overview of the service activities they contain, it lacks the details for designing a PSS specially the relations between needs, services and (physical) PSS components.

Lim et al. (2012) analyse the modelling techniques used for visualisation of PSS. They divide methods based on the aspect of PSS which is modelled. According to their study, most research focuses on the service processes of a PSS which among them, service blueprint got more attention for visualisation. Another studies aim at modelling the stakeholders of a PSS focusing on the relations between them, proposing alternative presentations of the service processes using a matrix called PSS board which shows how the PSS provider and the partners works to fulfil the customers' need in different stages of the service process. Maussang et al. (2009) argue that there is a gap between product development and the need for technical specifications of physical objects and the system approach. In order to close this gap, they suggest to use the graph of inter-actors and functional block diagrams for designing a PSS. They argue that functional block diagram is a useful tool for PSS modelling and analysis as functional representation of a PSS during its conceptual design phase is necessary.

According to Van Halen et al. (2005), appropriate tools are required to deal with high complexity of a typical PSS. They propose a methodology that offers a wide range of modelling methods for strategic analysis in different phases of PSS development. With several papers published in this area, we find that model integration is a common concept in the systems engineering process. However, most work in this area is on a rather high level, analysing the suitability of certain integration strategies, i.e. vertical vs. horizontal (Frank et al., 2014). Although there are approaches that utilize ontologies for modelling the dependencies between the various components of a PSS (Hajimohammadi et al., 2017), they remain at a high level of abstraction, thus suggesting that existing products and services are bundled together in order to form a PSS. In contrast to this, our approach is able to capture PSS where product and service components are engineered from scratch or at least modified in order to be integrated seamlessly.

2.2. PSS computer aided modelling tools

Several studies which reviewed the state of art of PSS engineering, discuss that there is need for software tools to support the modelling of PSS (Baines et al., 2007; Meier et al., 2010). Beuren et al. (2013) emphasize the need for tools that provide visualisation and modelling of different components of PSS including tangible and intangible elements to improve the understanding of a PSS engineering project. Morelli (2006) highlights the importance of a graphic representation technique for modelling PSS requirements. He also claims that while there are plenty of graphical notations in information sciences, they cannot be used for representing all elements involved in a PSS, like space, time and physical outlines.

Following this view, Sakao et al. (2009), discuss that a variety of the tools available for product development concentrate on physical and domain-specific details, but there is no particular tool that aims at designing integrated systems of services and products simultaneously. In order to close this gap, they propose a tool, called Service Explorer, which supports designing services according to value created by products' functions and user requirements. Service Explorer provides several modelling techniques, a database for managing services and some reasoning engines to help developers.

Komoto and Tomiyama (2008) argue that Service Explorer (Sakao et al., 2009) cannot explicitly elaborate the relations between services and products which is required for designing a PSS. They present a tool which combines service modelling with a life cycle simulator. The tool allows to analyse alternative PSS designs by quantitatively calculating economic and environmental performances from a holistic viewpoint. A relatively similar approach has been presented by Nemoto et al. (2015). They present a framework and software tool that allows to formalize design knowledge from previous engineering projects and existing PSS and use those insights for configuring a new PSS offering, but on a rather high level of abstraction.

2.3. Implications for comprehensive PSS engineering tool support

The works we discussed are mostly from the service design view and lack consideration of physical elements in modelling. However, we argue that there is no single modelling technique for development of a PSS that tackles the issues sufficiently. Since PSS are rather complex socio-technical systems, many different perspectives are required to be investigated for a comprehensive design.

The need for a tool that supports engineers in analysing the dependencies among artefacts has been recognized for a plethora of different use cases, e.g. for building the links between requirements engineering and safety analysis (Vilela et al., 2017) or integrating product life cycle management with service life cycle management of a PSS offering (Wiesner et al., 2015). Also Tang et al. (2007) argue that for complex software systems there needs to be traceability from rationale to design. The same is even more important for PSS were an even bigger picture needs to be taken into account. It is however surprising that the need for an automated traceability tool has been recognized more than 3 decades ago (Dorfman, 1984) and that still today satisfying solutions to this problem are scarce. In fact, a recent literature review on requirements engineering for PSS listed conflict detection and resolution among requirements, dynamic requirement change forecasting smart requirement management and proactive response as the main challenges to be tackled in the future (Song, 2017). Therefore, we aim at enabling an inclusive view by supporting visualisation of traceability links between different components of a PSS in a model integration ontology.

3. Model integration

As discussed in Section 1.1, PSS development involves various stakeholders which rely on their domain-specific models, diagrams or other development artefacts. Therefore, every artefact is created using specialised software tools, is specified in a domain-specific modelling language (DSML) and is serialised as one of many different persistent data formats. Besides, domain-specific models represent knowledge only from a particular perspective and just a fraction of the system is captured. Consequently, a thorough understanding of the comprehensive system design and the cross-domain dependencies is missing.

One way to address this problem is to use a single modelling language across all engineering domains involved. However, this approach comes with a huge disadvantage. The more concepts and logic a modelling language is capable of expressing, the more complex and complicated to comprehend it gets. Thus, establishing a single comprehensive modelling method covering all different aspects of a PSS leads to a complex and confusing representation (Eisenbart et al., 2012).

On the other hand, analysis of a PSS using models from different perspectives lacks consideration of relations between these models since many elements are interdependent. According to Chen et al. (2008), the lack of interoperability across the different departments of an enterprise is caused by interoperability barriers on four different levels (data, service, process and business). In our approach we focus on overcoming the issue of interoperability of data, since this is the fundamental layer for the integration of PSS components across conceptual, technological and organisational barriers. However, we encourage others the advance our work in order to support interoperability on a higher level.

We address the interoperability issue on the data level by proposing an integration ontology which specifies the generic artefact types (entities) that are used by the various DSML as well as the types of semantic relationships that can exist between those artefacts. This framework enables stakeholders with separate perspectives to analyse the relationships between their models and the others'.

To integrate models in one cross-domain representation, transformations from different domain-specific models are required. Thus, first we discuss briefly what types of model transformations exist; later the approach of this work for integration of different models is presented.

3.1. Model transformation

Since in practice, most of the models are graph-based or can be transformed into a graph (also natural language expressions can be viewed as a graph), we analyse mainly the graph-based transformations. In general, several types of graph-based model-to-model transformation methods exist. We categorise these methods based on two general criteria and explain each category separately.

Graph-based transformation methods can be direct or indirect based on whether models are transformed directly to each other or an intermediate model is applied. If the transformation mechanism requires both source and target models to be in the same technical space, we call it syntax-dependent transformation. On the other side, syntax-independent transformations are not based on a particular language. With regards to the introduced criteria, we classify all graph-based transformations into four categories: (Table 1).

Direct transformation methods define a set of rules which transfer source model to target model without use of an intermediate model. Direct syntax-dependent transformations requires both source and target models to use the same technical space (e.g. XML). In contrast, direct syntax-independent methods are not coupled to a particular technical space. This type of transformations first parses the source model into the syntax of target model's technical space, then map the parsed model to the structure conforming to the target model's metamodel (Mens and Van Gorp, 2006). While direct syntax-dependent methods can enclose the maximal possible transmittable detail from the source model to the target model (Varró and Pataricza, 2004), such methods are very limited regarding the languages which can be supported due to their binding to a concrete syntax. Syntax-independent direct transformations resolve this issue by decoupling the transformation rules from the technical space of the involved languages. For every pair of source and target models, direct transformations requires an explicit implementation. For example, Medvidovic et al. (2003) present a model integration approach that uses direct mappings from one DSML to another. However, this approach is primarily focused on syntactical issues than on semantics (meaning) of the relationships between artefact and it requires individual mappers for each pair of modelling languages that is to be covered. This problem has been addressed by indirect transformations.

Table 1
General types of model transformations.

Direct Syntax-dependent	Direct Syntax-independent
Fixed Intermediate language	Flexible Intermediate language

Indirect transformations rely on a well defined intermediate language and each transformation process involves transforming from the source model to the intermediate language and afterwards, transforming from intermediate language to the target model (Czarnecki and Helsen, 2006). We refer to a syntax-dependent indirect transformation as fixed intermediate language transformation and similarly syntax-independent indirect transformation as flexible intermediate language transformation. The intermediate language of a fixed method is defined in the same technical space of the source and the target models. Flexible indirect methods are not bound to the underlying syntax of a technical space, but rather the intermediate language is only expressed in its concepts. Thus, with higher level of abstraction, the intermediate language can support more models and it is more flexible regarding future adaptations (Bézivin et al., 2006). Here, The great advantage is that one does not require a separate set of transformation rules between every two DSML but only between each DSML and the intermediate language.

4. TRAILS integration method

In its core, TRAILS is founded on two basic concepts, model integration and model transformation with the latter being a prerequisite for the first. In a nutshell, the basic ideas behind TRAILS is to interrelate all available engineering information within a semantic graph by transforming various kinds of domain-specific models and other engineering artefacts into a format that conforms to a cross-domain model integration ontology. We explain these basic concepts in the following in order to lay a solid ground for further explaining the core features of TRAILS.

4.1. Model integration ontology

As presented in detail in some of our earlier publications (Kernschmidt et al., 2013; Wolfenstetter et al., 2014; 2015), we developed an ontology of PSS engineering artefacts whose evolution needs to be captured in order to ensure traceability. However, this ontology is not solely focused on ensuring traceability but also on the more generic issue of integrating engineering information which is an essential prerequisite for ensuring traceability. In this sense, it defines the fundamental ontological concepts in the context of PSS development and during service provision, such as requirements, actors, business processes or decisions made in the engineering process. Additionally it specifies which types of semantic relationships can exist between those ontological concepts. For example, a solution component *satisfies* a requirement or an actor *performs* an activity. For the purpose of merging the engineering knowledge distributed over several domain-specific models, every artefact that is imported into TRAILS is translated to comply with this model integration ontology. This means that the domain-specific ontologies that are defined by the meta-models of the DSML are mapped onto a common language using the proposed ontology as a meta-model. In the following we explain the structure of the model integration ontology in detail.

The model integration ontology uses on the most abstract level three basic *Elements* to describe the artefacts of PSS engineering and their dependencies. These *Elements* are *Nodes, Edges* and Attributes with each of them being hierarchically further decomposed into more concrete types of ontological concepts.

The general purpose of *Attributes* is to capture descriptive information such as duration, weight, price or colour of ontological entities as well as meta-information (e.g. name, id, date of creation etc.); basically everything that cannot be considered as an entity itself. Naturally, attributes can have attributes themselves, for example names or units. Furthermore, it is possible to define *Relationships* between *Attributes* to e.g. define the mechanics of unit conversions.

Regarding *Edges* the TRAILS model integration ontology differentiates between *Flows* and *Relationships*. On the one hand, *Flows* characterise the transferral or transmission of value, material, energy or information between two entities or describe the order of activities in a process, i.e. *Control Flow*. On the other hand, there is the concept of *Relationships* which are universally valid while *Flows* have a cause and are bound to a defined period of time in which they occur. In this context, the TRAILS integration ontology distinguishes *Causal Relationships* (e.g. create), *Chronologic Relationships* (e.g. evolves to), *Referential Relationships* (e.g. refers to), *Inclusion Relationships* (e.g. part of) and *Inheritance Relationships*.

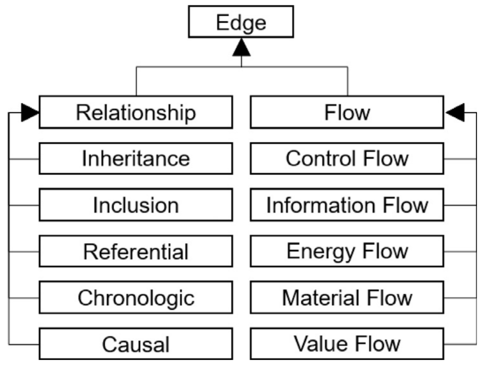


Fig. 1. TRAILS Model Integration Ontology: Edge Types.

As with *Nodes*, we further differentiate between two generic types (Fig. 2). *Solution Artefacts* represent the solution to the original customer problem, i.e. the features, behaviour and (component) structure of the PSS itself. *Development Artefacts* specify the problem domain and the process of working on a solution to these problems, i.e. the development process. Additionally, since PSS are socio-technical systems, humans that interact with the PSS or are by other means related to the development of the PSS or to service provision are summarised as stakeholders and on more concrete levels of the ontology decomposed into the various sub-types.

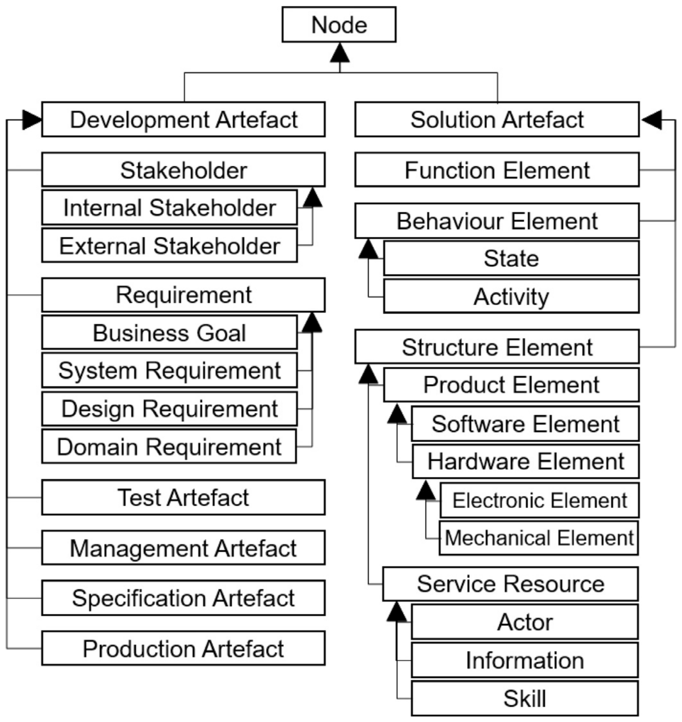


Fig. 2. TRAILS Model Integration Ontology: Node Types.

As discussed, *Development Artefacts* are supposed to contain information related to the development and they are not part of the resulting PSS. On a more detailed level, we distinguish between five different sub-types of *Development Artefacts* in PSS engineering. First, *Requirement Artefacts* are used to structure and define the problem for which the final PSS represents a solution.

In the context of PSS engineering, requirements can be broken down into four levels of abstraction. On the highest level, business goals of the PSS are being defined. Based on these business goals, system level requirements are derived which represent for example the needs of stakeholders, environmental considerations and business process demands. In the next step, design level requirements are elicited to address the details of system level requirements. Again, at the most detailed abstraction level the design level requirements are translated into domain-specific requirements which all different involved domains (e.g. software engineering, mechanical engineering or service engineering) can work with.

Specification Artefacts are means to describe requirements or system designs by using different techniques. *Specification Artefacts* can take various forms. Most commonly they appear as natural language texts, graph-based models or other sorts of diagrams. However, also sketch drawings, other kinds of illustrations and even videos could serve as *Specification Artefacts*.

Test Artefacts are any kind of artefact that serve in the process of checking whether the solution satisfies the requirements. Hence, the variety of potential *Test Artefacts* ranges from mathematical or logical proofs to computational simulations, experiments to informal methods, such as stakeholders' feedbacks.

Production Artefacts capture and represent knowledge that is relevant for the manufacturing process of hardware components of the PSS. This includes for example, the machinery within the assembly line, equipment used during the process or the specification of the manufacturing process itself. This way it is possible to link physical PSS components to the manufacturing process and trace whether changes to the design of hardware components impact the set-up for component manufacturing.

Management Artefacts keep track of issues initiated in the development of a PSS. For example a change request or a decision information are considered as *Management Artefacts*. They are predominantly of use when tracing the evolution of other types of *Development Artefacts* over time.

Stakeholders represent different types of roles who are involved in the PSS life cycle. Handling complex network of stakeholders in a PSS is challenging due to several reasons. Since a PSS requires a new business model, the entire organization of PSS provider is affected and consequently new requirements for each department should be addressed. Besides, by adding services to a core product, new range of stakeholders will be included in development of a PSS. In the ontology we distinguish between two general types of stakeholders: *Internal Stakeholders* and *External Stakeholders*. Internal stakeholders refer to all units and collaborators to providing the PSS. In contrast, external stakeholders are not part of the PSS providing organization.

Solution Artefacts refer to components which construct the PSS including products and services. In the meta-model we classified *Solution Artefacts* into *Function Elements, Behaviour Elements* and *Structure Elements. Solution Artefacts* satisfy *Requirement Artefacts* and they are verified by *Test Artefacts*.

Structure Elements are fundamental resources constituting a PSS which are categorized to *Product Elements* and *Service Elements*. Both tangible resources, like material, and intangible resources, like information, contribute to structure elements. The system performs some workflows and processes in order to accomplish a target function. These are presented in the *Behaviour Element* and the functions of the system are presented in *Function Element*.

4.2. Model transformation process

To consolidate the engineering information that is contained in the various domain-specific models, TRAILS transforms these models into a semantic graph that conforms to the model integration ontology presented in the preceding section. In order to support model transformations between different technical spaces and serialization formats, the model transformation process itself is divided into two independent steps. First, so-called *I/O mappers* are used to de-serialize various data formats, such as ReqIF, XMI or even CSV, and representing them as a generic graph structure. In this context, generic graph representation means that the resulting data structure consists of only untyped nodes, edges and attributes. At this point, the concrete syntax that is used by domain-specific modelling and engineering tools has been transformed into an abstract syntax that still conforms to the meta-model of the source DSML. As a second step, *model mappers* transform the generic graph into a semantic graph that conforms to the specifications defined by the model integration ontology. To ensure a high level of transformation accuracy, TRAILS uses customized model mappers for each domain-specific modelling language. However the general mode of operation is similar for each model mapper.

Each *model mapper* contains a set of rules that specify an equivalent for every node, edge or attribute of the respective meta-model within the model integration ontology. In the most simple case, each element can be mapped on one equivalent element of the same type (e.g. a node type being mapped to another node type). However, in other cases mapping patterns are more complex. For instance, a set consisting of two nodes that are linked by a certain type of edge could be mapped to one single node with a certain attribute. Overall, each model mapper uses a sequence of atomic transformation operators illustrated in Fig. 3 to perform the transformation process.

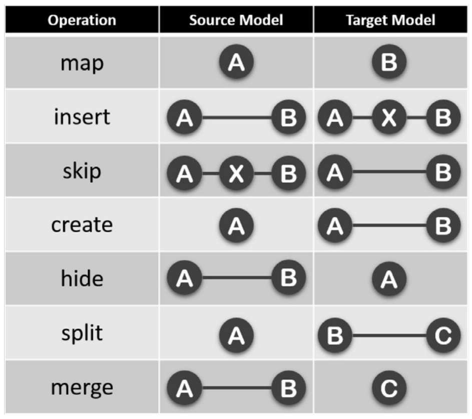


Fig. 3. Generic Transformation Operators.

Probably the most intuitive transformation operator is to *map* one type of element from the source model to an analogical element of the target model, i.e. edge to edge, node to node and attribute to attribute. In this case, the mapping operator works bi-directionally and each element can be considered as the equivalent of the other. Simply speaking, this operation does merely just change the name of corresponding type to the one defined in the target model. Of course, performed on nodes and edges, this operation entails analogous operations on their attributes as well.

The *insert* transformation operator is mainly performed on two nodes that are connected by an edge. For a certain pattern (node A connected by edge 1 to node B) this operator splits the edge into two and inserts a predefined node X at the junction. Amongst others, this operator is used for in cases were the meta-model of a DSML would not allow a direct edge between two nodes. For example, the Event-driven Process Chain (EPC) meta-model does not allow a control flow from one activity to another, but only between activities and event. So when transforming a UML activity diagram via the model integration ontology to an EPC model, additional event nodes need to be inserted. As depicted in Fig. 3, there is also an analogous *skip* operator that performs the inverse of the *insert* operator. This means, when detecting a node X that is connected to A and B via an edge of type 1 it removes X and links A and B directly.

When transforming models, it is often the case that for certain elements in the source model their equivalent in the target model misses important attributes. In this context, it is often necessary to *create* an additional node to capture the information that would otherwise get lost. Also, some modelling languages allot that certain nodes only appear in connection with other nodes. For example, in UML use case diagrams, use case always need to be directly or indirectly linked to actors. Vice versa, the *hide* operator ignores for example nodes in the source model that are not intended in the target model. Wherever possible, it preserves the information contained in the discarded node by adding it to the one that is maintained.

Another inverse pair of atomic transformation operators are used to *split* or *merge* nodes. When specified for a certain type of node within the source model the *split* operator creates two separate nodes of another type than the original node connected by a pre-defined edge. This kind of operator is required usually if an element in the source model has no direct equivalent in the target model. In this case, after the transformation the information that was before carried by one node is split onto two separate nodes. An example for this approach can be found when transforming a BPMN diagram into a format compliant to the model integration ontology. While gateways in BPMN contain the decision that is being made as well as the logical consequence (i.e. which control flow is being followed after the decision) the concept specified differently in the integration ontology. Here, the activity and the logical conjunction (AND, OR, XOR) are treated as two separate nodes. Therefore, when importing a BPMN diagram TRAILS will apply the *split* operator and vice versa, the *merge* operator when exporting to BPMN.

By separating the model transformation process into I/O mapping and model mapping each I/O mapper can be used for transforming various modelling languages or meta-models respectively. Correspondingly, each model mapper is able to transform a specific type of meta-model into a graph representation that conforms to the model integration ontology for PSS that is used in TRAILS. Apart from importing various domain-specific models, TRAILS is also capable of exporting the integrated PSS engineering information into DSML and data formats. For this purpose, the model transformation process, as described before is reversed.

5. TRAILS Features

TRAILS pivotal mission is to support various stakeholders along the PSS life cycle in integrating, analysing and enhancing the knowledge that is often spread across and hidden in the multiple different engineering artefacts. The tool therefore provides a number of features related to import, merging, editing and analysing graph-based models from various engineering domains.

5.1. Importing models

The core ability of TRAILS is that different types of model specification formats can be imported and transformed into the cross-disciplinary representation defined by the TRAILS Model Integration Ontology. Furthermore, the entire semantic graph that results from integrating the various types of PSS engineering artefacts can later be entirely or in parts transformed into other formats supported by the tool.

One type of specification formats supported by TRAILS is for example the rather text-oriented *Requirements Interchange Format*(ReqIF). ReqIF is a format based on XML which enables stakeholders with different modelling and requirement authoring tools to collaborate by exchanging their requirements' information.

Since PSS intend to be solutions to specific needs it is crucial for the PSS provider to fulfil the customer's requirements as complete as possible. Requirements engineering is thus one of the most important activities both during PSS development as well as service provision. Due to the dynamic business environment in which PSS compete requirements eventually change over time and the PSS needs to be adapted accordingly. By linking the information that is contained in requirements documents to system design models of the product components or process models related to service delivery it is possible to anticipate the impact of changing requirements on the PSS design more accurately and anticipate the consequences. The information that is needed in this context can be imported from distinct requirements engineering tools using the ReqIF format.

Two other important modelling languages that are relevant in PSS engineering are the *Unified Modelling Language* (UML) and its almost twin brother, the *Systems Modelling Language* (SysML). UML, the older of both brothers, originated from software engineering and was developed to provide a common platform for system architects and software developers to communicate over system analysis and implementation.

When recognizing the advantages of a notation that allows to logically decompose complex systems, UML subsequently entered the mechanical engineering domain and was adapted to fit its characteristics. The resulting modelling language SysML extends a subset of the basic UML diagram types but also introduces new concepts, such as ports. So, while UML is a more software-oriented modelling language, SysML aims at modelling and designing complex systems that rather stem from the mechanical engineering domain. However, both of them offer various diagrams for specifying a systems structure as well as its dynamics.

While UML and SysML can be used to specify the rather technical aspects of a PSS, namely the hardware and software components, the service engineering domain mostly relies on notations to specify business processes. Widely used modelling techniques for business processes are the *Business Process Modelling Notation* (BPMN) as well as *Event-driven Process Chains* (EPC). An EPC, for example, is an ordered graph of events and functions that enables describing alternative and parallel execution of processes and it is enhanced with logical operators like AND, OR, etc. The structure and notation of both, BPMN and EPCs, is very similar and they are often supported by the same software tools, e.g. MS Visio. We thus chose this software tool as an example to show the import of such process models.

Although they are not commonly referred to as modelling languages, TRAILS supports the import of (and export to) other important formats. For example, the *Resource Description Framework* (RDF) is a general technique for conceptual description of resources. It is widely used in the context of semantic web applications for specifying entities and their semantic relationships to each other forming an ontological graph that explains real world concepts. TRAILS uses the RDF format to define the structure of the model integration ontology it uses internally as a model representation format. Furthermore, it is used to define the model mapping rules that are applied when importing and exporting models in another language. RDF models can be serialized in various formats, the most important being probably the *RDF-XML* format and the *Terse RDF Triple Language* (TTL) which is a serialization format that is easier for humans to read than the widely used RDF-XML format. Since RDF is a technology that is used across various domains in order to structure and represent knowledge in a graph-based form, we chose to support both formats in TRAILS.

5.2. Merging models

Since PSS aim to be customer-centric solutions in which the individual components need to be integrated seamlessly to provide the desired service and guarantee a enhanced customer experience, developers need to be able to evaluate how the design and the behaviour of the individual components impacts each other. In order to do so, it is helpful to identify and explicitly model the dependencies and overlaps among the various domain specific development artefacts involved in the PSS development.

For this purpose, TRAILS allows to merge the various models of a PSS, each describing a specific viewpoint on the system as a whole, by identifying common concepts or entities in the different models. After importing the different domain-specific models into TRAILS they are merged into a semantic graph with the nodes of this graph representing entities or real world concepts, respectively.

Each time a new model is imported, TRAILS can perform model merging to determine the overlaps of the newly imported domain-specific model with the integrated model in the database. In order to identify model overlaps, i.e. similar nodes or sub graphs, TRAILS uses three types of similarity calculation methods, each consisting of multiple approaches. First, model overlaps can be determined by calculating the similarity of the descriptions or captions of model elements. Examples of such approaches are String Edit Distance or Levenshtein Distance that reflect how similar the captions of two model elements are. Second, TRAILS is able to determine the similarity of model elements by evaluating their attributes. In general, model elements that have some identical attributes tend to be similar or at least closely related. And third, TRAILS uses a method we call context similarity evaluation. This methods determines the similarity of two nodes based on the similarity of their neighbours. According to this method, two nodes have a high similarity if their adjacent nodes in the graph appear to have the same type, name or other attributes. In model-based engineering it is reasonable to expect that model elements with similar adjacencies are closely related or identical.

The different model similarity indicators can be combined flexibly to allow for optimal merging results. TRAILS then presents the results of the similarity calculation to the user ordered by a combined similarity measure. For each pair of likely similar elements the user can the select to merge two nodes into one, link the nodes using an edge that describes their relationship (e.g. new version of) or ignore the similarity. By offering comparison algorithms that can be combined flexibly, TRAILS provides the means to implement automated procedures for traceability maintenance as proposed by Mäder and Orlena (2011) when it comes to changes along the engineering life cycle.

5.3. Adaptable cross-domain model integration ontology

As stated before, when importing models from external software tools that are described in a certain DSML or format, TRAILS maps those imported models onto a cross-domain ontology (cf. Section 4.1) that has the expressive power to integrate the viewpoints of multiple engineering domains.

Although the TRAILS model integration ontology incorporates concepts from various DSML, it is not feasible to consider every modelling language or format ever invented. In fact, more or less every company uses some self-developed legacy software tools or data formats that they customized to their needs. They vary from customized and extended off-the-shelf engineering tools to simple spreadsheets enhanced by using macros.

In order to deal with the issue of having to interoperate with non-standard software tools and data formats, the TRAILS integration ontology can be modified and extended by the user according to individual, context-dependent needs. This feature allows the ontology to be adapted to specific needs of the application environment, such as specific industry or project characteristics. This way, it is also possible define additional ontological concepts that are needed in order to import artefacts which are specified in further modelling languages or data formats that are not part of TRAILS standard implementation. However, if the integration ontology is altered, in some cases rules for model mapping have to be adapted as well. In TRAILS, the integration ontology as well as mapping rules can be accessed, managed and modified directly within the graphical user interface. Furthermore, the files containing this information can be exchanged with other users.

5.4. Editing models

Although TRAILS supports the user in producing an integrated model of the PSS from a number of different sources by offering smart merging algorithms, in some cases there is need for manual editing. This way, the user is able to add missing links or clean up duplicate information, both of which are very likely to cause inconsistencies within the product design and specification. Again, in some cases such inconsistencies ultimately lead to product failures and costly callback if they remain undetected.

Furthermore, it is possible to create new nodes, add them to the integrated PSS model and complement or adjust attributes. Hence, the user can add details to the system specification that could not have been expressed in the source modelling tools. By doing so, the user is able to enrich the system model by adding additional information that is required for e.g. change management or requirements tracing. Thus, instead of limiting itself to just importing, processing and interpreting data that has been created using other software tools TRAILS allows the user to edit or delete the nodes and edges that the imported models consist of.

5.5. Customisable appearance and standard graph layouts

The fundamental idea of TRAILS is to structure and illustrate the entire knowledge about the product or solution being developed and the development process itself as a semantic graph. This graph consists of nodes which represent any kind of artefact created in the engineering process as well as edges which represent different types of relationships among these artefacts.

As a consequence, the visual appearance of this graph determines the comprehensibility of the model and consequentially the usability of the TRAILS software tool. PSS engineers not only need to understand the increasingly complex technical products that are part of the PSS solution but also the dynamics of the business processes in which the service is provided to the customer. Therefore, intuitive presentation of engineering information and the possibility to interact with the semantic graph are crucial features. This way, TRAILS supports the user in revealing hidden information through rearranging the graph or highlighting certain nodes or edges.

Specifically, TRAILS allows to automatically arrange the displayed graph in one of several pre-defined standard graph drawing strategies, such as circular or force-based layouts. Furthermore, the user can rearrange nodes manually and expand or compress nodes that contain sub-graphs.

In addition to that, the tool offers the possibility to customize the appearance of nodes and edges. The user can select the standard colours for each type of node and edge in order to facilitate visual differentiation. Moreover, it is possible to chose between multiple node shapes or embed individual icons or images for each type of node.

5.6. Customised filtering and viewpoint creation

The analysis of integrated engineering information as it can be performed using TRAILS is in some ways a double-edged sword. On the one hand it is desirable to collect and integrate as much information as possible from various domain-specific models, documents and other sources. On the other hand, one quickly obtains a tremendously complex network of interrelated artefacts that in all its details is hardly comprehensible for the human analyst at first sight.

However, most analysis tasks only concern a minor fraction of the nodes and edges that form the integrated PSS model. In addition to various graph layout options, TRAILS provides a customized filtering feature. This feature allows the user to filter the graph according to node types, edge types and even attribute values in order to decrease the amount of information visualised to what is actually needed for performing the analysis. With the possibility to only display a certain fraction of the nodes and edges that form the graph the user is better able to e.g. follow the evolution of a particular requirement over time or oversee just the information flows within the PSS.

In TRAILS these customized filters are referred to as viewpoints on the semantic graph. These viewpoints can be defined manually by the user or loaded from pre-defined templates that serve specific purposes or reflect a certain role in the engineering process, e.g. the requirements analyst. Once defined, a viewpoint can be saved to the viewpoint selection for future use. In addition to simplifying the visual appearance of the integrated PSS model, viewpoints allow to restrict access for certain roles, e.g. when engineering information needs to be shared with external stakeholders.

5.7. Matrix view and spreadsheet integration

In some engineering disciplines like mechanical or industrial engineering matrix-based engineering tools such as design structure matrices (DSM) or domain mapping matrices (DMM) are continuously popular. Although these tools are often naturally grown legacy systems, they are widely used across various industries and most companies use at least one tool of this kind in their engineering process.

For some use cases, like generating supplementary nodes or capturing a larger number of traceability relationships, matrices constitute a more convenient form of visualisation. Using matrices a larger number of nodes can be arranged in a space-saving manner allowing for a more compact overview of the overall system structure. Hence, besides the default graph view, TRAILS also provides the possibility to visualize the semantic traceability graph as a matrix. Similar to the graph view, TRAILS allows customised filtering and viewpoints in the matrix view as well. Moreover, all graph editing operation can be performed the same way using the matrix view. This way, the user is able to switch between both forms of visualisation flexibly using the perspective the fits best for the respective task (Tilstra et al., 2010).

Another reason why many companies still use such matrix-based engineering tools is that they can be implemented using standard office software for spreadsheet calculation. This way, matrix-based engineering tools can be flexibly used and easily adapted. In order to ease integration with such tools, TRAILS allows to export the semantic traceability graph or parts thereof to common spreadsheet formats so that the data can be analysed using existing matrix-based analysis tools.

5.8. Multi-user capabilities and database server

As technical products become more and more complex, so have the processes of their development. Today, the development of technical products usually involves a team of specialists from multiple engineering domains to design and integrate the various components, i.e. hardware, software and in many cases services. Hence, the typical engineering process and with it engineering software tools are increasingly shaped by the need for collaborative and concurrent team activities.

The need for supporting collaborative engineering through adequate tools is even more prominent when multiple companies are involved in solution design and service delivery. The various stakeholders from different companies sometimes distributed globally need to be able to work concurrently on a central instance that serves as a single point of truth for the integrated PSS model.

For this purpose, TRAILS provides the possibility to store all engineering data on a central graph database server. In our current implementation we use the open-source framework Apache Jena as database to store RDF data together with Fuseki Server for serving RDF data over standard internet protocols, such as HTTP. This way, multiple installations of TRAILS can be used concurrently and synchronise updates with the central database.

6. Case study: Bike sharing system

In this section, we explain a bike sharing system situation as a PSS example to clarify how TRAILS can support PSS development. The bike sharing system is not a hypothetical example but was implemented as a functioning prototype at our university with multiple departments collaborating extensively. Overall, service processes where designed, software components (such as central database systems and a mobile application) where implemented and the necessary modifications to a bike where engineered and built in order to set up a functional prototype of a bike that operates within a free floating bike sharing system. This way, it was possible to have control and unlimited access to the whole PSS engineering process which would not have been the case in a real industry example. Even though TRAILS was evaluated in a much more detailed case study than can be presented here, we admit that several industry case studies are desirable in order to evaluate the true potential of our software tool. Nonetheless, we think that a simplified version of our case study gives the reader a better idea of what TRAILS is and how it functions.

In order to understandably present our case we only describe the high level architecture of the system. In this conjunction, our goal is not to present every feature of TRAILS in detail, but to give the reader a general overview of the idea behind our tool using an intuitive application scenario. Bike sharing systems are a typical example of a PSS with the aim of providing mobility as a service to customers. In our case study the PSS provider offers bikes on an on-demand basis to customers at multiple sharing stations within the city limits. The bikes can be rented by registered customers simply by entering their customer ID and PIN at one of the bike sharing stations and one of the bikes would be released. The customer can then use it on a pay-per-minute basis and then return it at the same or any other of the bike sharing stations. After returning the bike, the amount due is charged to the customers credit card.

From an architectural perspective, the bike sharing system is composed of stations that feature a keyboard and screen as a user interface. Besides, stations are equipped with an external power supply and they communicate with the back-office fleet management system using a mobile telecommunications module (UMTS module). Bikes can be locked to the stations via an electric lock. This lock is released when the customer rents one of the bikes. The back-office fleet management system, which operates in the background, is further linked to a central database as well as a payment system (Fig. 4).

As in a PSS we are not only dealing with products, but also services, to achieve a full understanding of the bike sharing system, it is necessary to consider the dynamic service processes (Fig. 5). When a bike is needed, the customer walks to the next sharing station and enters his user ID into the user interface. The sharing then checks the user ID for validity by communicating with the central database. If the user ID is valid, the system will change to the next screen, where the customer can enter his PIN, which again will be checked for validity in the database. If this PIN is also correct, the sharing station then releases one of the bikes by opening its electric lock.

We consider the case that the bike sharing provider re-investigates market trends as well as technological possibilities in order to improve its customer service and reach out to new market segments. The bike sharing provider reaches the conclusion that it should change its business model from offering stationary bike sharing to offering a free-floating system where customers can pick up or leave the bike anywhere in the city area. Fig. 6 depicts a requirements document containing selected requirements for the free-floating system that influence some of the existing structural components and services processes which need to be adapted according to the new business model. In order to satisfy these requirements, architecture of the bike sharing system needs to be adapted. As the sharing system no longer depends on fixed stations, there has to be an alternative way for customers to locate the nearest available bike. In the new system architecture a smartphone

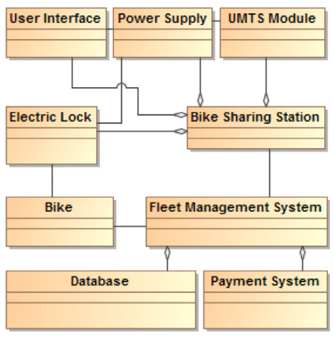


Fig. 4. SysML block diagram of the stationary bike sharing system.

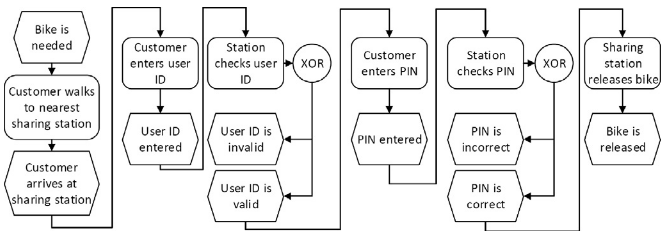
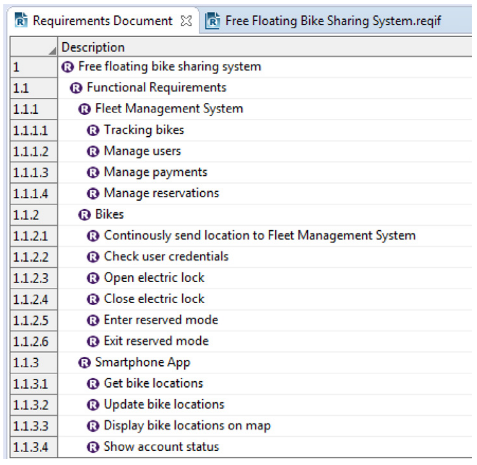


Fig. 5. EPC of the rental process in the stationary bike sharing system.



Requirements Document ☒ Free Floating Bike Sharing System.reqif

	Description
1	Free floating bike sharing system
1.1	Functional Requirements
1.1.1	Fleet Management System
1.1.1.1	Tracking bikes
1.1.1.2	Manage users
1.1.1.3	Manage payments
1.1.1.4	Manage reservations
1.1.2	Bikes
1.1.2.1	Continously send location to Fleet Management System
1.1.2.2	Check user credentials
1.1.2.3	Open electric lock
1.1.2.4	Close electric lock
1.1.2.5	Enter reserved mode
1.1.2.6	Exit reserved mode
1.1.3	Smartphone App
1.1.3.1	Get bike locations
1.1.3.2	Update bike locations
1.1.3.3	Display bike locations on map
1.1.3.4	Show account status

Fig. 6. Excerpt of requirements document for free floating bike sharing system.

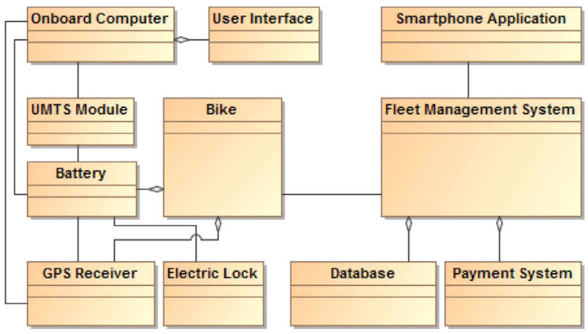


Fig. 7. SysML block diagram of the free floating bike sharing system.

application that customers can install on their device will display the location of bikes on a map. This smartphone application needs to be able to communicate with the central fleet management system, which will store and continuously update the position and availability of nearby bikes. Furthermore, the user interface and electric lock that used to be part of the fixed sharing stations now need to be integrated into the bikes themselves. In order to be able to communicate with the back-office fleet management system for updating the bikes location or checking the users credentials the bikes need to be equipped with a UMTS module as well as a GPS receiver. All of the electric equipment furthermore needs to be powered by a battery attached to the bike.

Apart from the system structure, the service processes need to be adapted as well. As an example, we again take the rental process (Fig. 8). This process has been adapted in order to fulfil the requirements of the new business model. If a bike is needed the customer first opens the bike sharing application on his smartphone, which then checks for nearby bikes and displays them on a map. The customer can then select one of those bikes and reserve it for the next few minutes, until he reaches the bike. A reservation request is directly sent to the fleet management system (FMS), which then updates the bikes status in the database and forwards the reservation request to the bike. When the customer then reaches the bike, he can proceed as normal by entering his user ID and PIN into the user interface attached to the bike. The credentials are sent to the fleet management system and if correct, finally the electric lock opens.

In order to record the changes of the system architecture as well as the rental process in TRAILS we need to capture the changes in all affected development artefacts, i.e. the SysML block diagrams, the EPC process models and the ReqIF requirements document. As we map the different versions onto each other and also link the altered requirements to the solution artefacts that fulfil those requirements, we can easily understand which parts of the system need to be adapted and

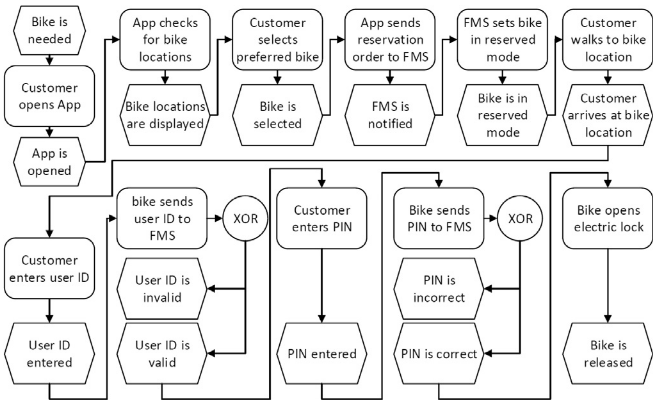
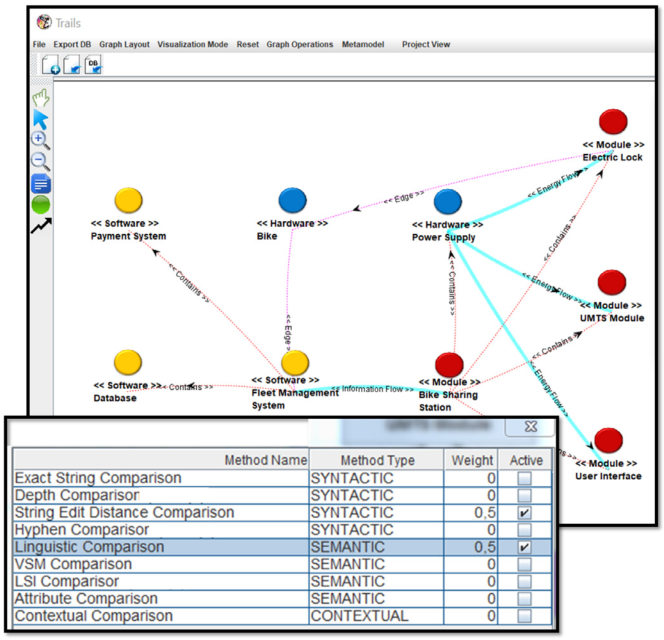


Fig. 8. EPC of the rental process in the free floating bike sharing system.

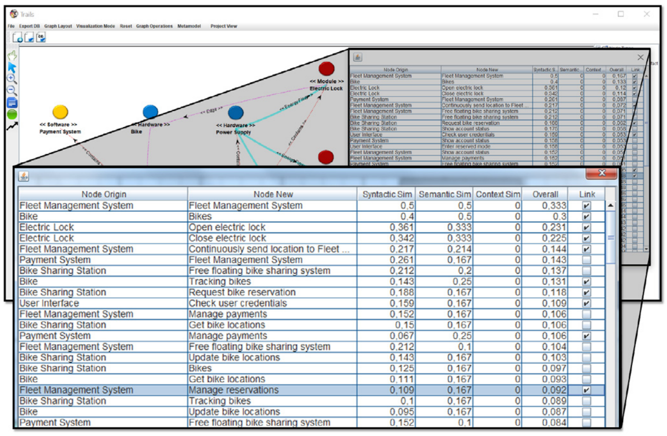
how. In order to do so, the different versions of the SysML block diagram need to be imported into TRAILS. When merging several source models into one comprehensive description of the system and its evolution, TRAILS offers several comparison algorithms that support the user in identifying model overlaps and common elements. Fig. 9 shows this step in the process of merging the block diagram of the stationary bike sharing system being merged with the altered requirements document.

At this stage, the user can select which of the comparison calculations (as introduced in Section 5.2) should be performed. Here, it is also possible to select multiple comparison algorithms and calculate a combined weighted similarity score. TRAILS then evaluates the similarity of elements within the two models that are being merged and displays them as a sortable list. The user may then decide which of the pairs of nodes should be linked or merged into one. The result of comparing the block diagram of the stationary bike sharing system with the requirements document can be seen in Fig. 10. For this merging process an equally weighted combination of vector space model comparison and string edit distance was chosen in order to identify similarly named entities. In this example we see that even though the algorithm has to operate on a very simplified requirements document that contains only captions and not the requirement descriptions themselves, we can easily identify solution artefacts and requirements that are related.



Method Name	Method Type	Weight	Active
Exact String Comparison	SYNTACTIC	0	☐
Depth Comparisor	SYNTACTIC	0	☐
String Edit Distance Comparison	SYNTACTIC	0,5	✓
Hyphen Comparisor	SYNTACTIC	0	☐
Linguistic Comparison	SEMANTIC	0,5	✓
VSM Comparison	SEMANTIC	0	☐
LSI Comparisor	SEMANTIC	0	☐
Attribute Comparison	SEMANTIC	0	☐
Contextual Comparison	CONTEXTUAL	0	☐

Fig. 9. Configuration of comparison algorithms for merging models.



Node Origin	Node New	Syntactic Sim	Semantic Sim	Context Sim	Overall	Link
Fleet Management System	Fleet Management System	0,5	0,5	0	0,333	✓
Bike	Bikes	0,4	0,5	0	0,3	✓
Electric Lock	Open electric lock	0,361	0,333	0	0,231	✓
Electric Lock	Close electric lock	0,342	0,333	0	0,225	✓
Fleet Management System	Continuously send location to Fleet ...	0,217	0,214	0	0,144	✓
Payment System	Fleet Management System	0,261	0,167	0	0,143	☐
Bike Sharing Station	Free floating bike sharing system	0,212	0,2	0	0,137	☐
Bike	Tracking bikes	0,143	0,25	0	0,131	✓
Bike Sharing Station	Request bike reservation	0,188	0,167	0	0,118	✓
User Interface	Check user credentials	0,159	0,167	0	0,109	✓
Fleet Management System	Manage payments	0,152	0,167	0	0,106	☐
Bike Sharing Station	Get bike locations	0,15	0,167	0	0,106	☐
Payment System	Manage payments	0,067	0,25	0	0,106	✓
Fleet Management System	Free floating bike sharing system	0,212	0,1	0	0,104	☐
Bike Sharing Station	Update bike locations	0,143	0,167	0	0,103	☐
Bike Sharing Station	Bikes	0,125	0,167	0	0,097	☐
Bike	Get bike locations	0,111	0,167	0	0,093	☐
Fleet Management System	Manage reservations	0,109	0,167	0	0,092	✓
Bike Sharing Station	Tracking bikes	0,1	0,167	0	0,089	☐
Bike	Update bike locations	0,095	0,167	0	0,087	☐
Payment System	Free floating bike sharing system	0,152	0,1	0	0,084	☐

Fig. 10. Merging Results of Requirements with Block Diagram for a Stationary Bike Sharing System.

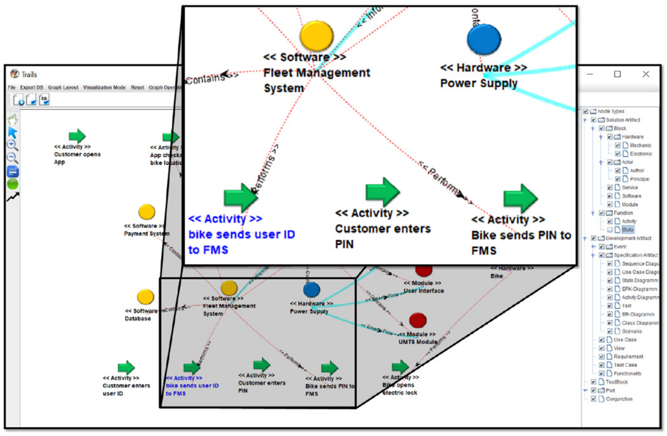


Fig. 12. Linking activities to system components.

Having merged the specification of the system architecture, TRAILS provides a visualisation of the combined model. This way, the user can explore the model visually in order to get a clearer understanding of the inherent system structure and complement missing links between related elements of the models. As illustrated in Fig. 11, TRAILS not only shows which of the requirements impact a certain solution component but also shows the internal break down structure of the requirements document. Consequently, the user is able to deduce the dependencies between individual requirements. In this context the user can furthermore chosen which types of semantic relationships to highlight or which to fade out, thus increasing clarity of the visualised dependencies.

To trace the changes regarding the system components that result from evolving the bike sharing business model from stationary to free float, a next step would be to merge the current system architecture as depicted in the SysML block diagram with the future architecture that was specified according to the new requirements. By doing so, engineers can see at first glance which components have been changed. However, changes not only manifest themselves in the architectural setup of the system, they also incur in the system behaviour, i.e. the business processes. Again, changes in the business processes may impact the system architecture. Thus, it is advantageous to visualise which components are involved in performing certain service processes. Fig. 12 shows the result of merging the SysML block diagram that reflects the structure of the PSS with the EPC diagram illustrating the bike rental process. Through visualisations like this, engineers can easily identify which components are likely to be impacted by a change in the business processes.

Although the models in our case study are only a rudimentary description of a PSS, they permit some insights into the traceability and model integration support that is offered by TRAILS. Once imported and merged into a comprehensive system model, TRAILS stores the resulting semantic graph in a central database that can be accessed by multiple clients. Throughout development new versions of the solution or development artefacts can be continuously merged with the existing model on the database. TRAILS will then constantly update the evolution of these artefacts and enable users to query the semantic graph in order to get new insights.

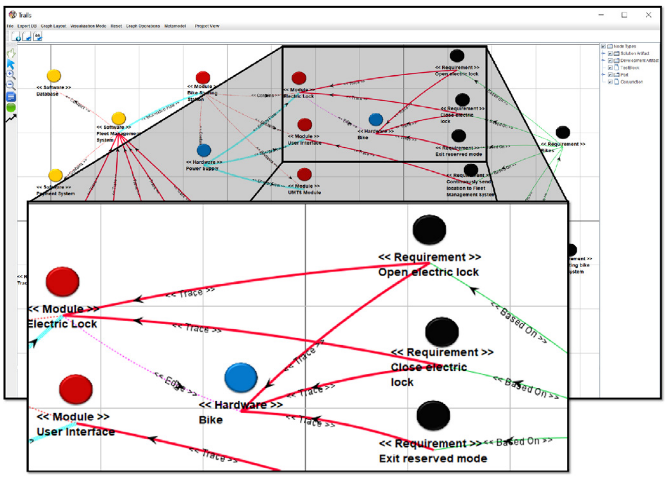


Fig. 11. Linking Requirements to Solution Components.

7. Discussion

As discussed in the first section of this paper, there is no tool that supports integration and holistic analysis of heterogeneous PSS engineering artefacts along with the dependencies between those. To address this issue, we developed a tool prototype that enables integrating models from different domains of PSS engineering, the visualisation of the relationships among the merged elements and managing the changes through a version management mechanism. In this section, we discuss the strengths and weaknesses of this tool prototype including its features and the methods and technologies employed.

Capturing of PSS artefacts in a semantic engineering graph: As its main functionality, TRAILS allows capturing the relationships between all types of artefacts like actors, use cases, decisions, process activities, product components and so on. The graph-based presentation enables easy understanding of the dependencies among an within artefacts. While most modelling tools only allow the user to view the relationships of element within a specific model and others like document management systems focus on the evolution of as well as the relationships between certain documents, TRAILS is able to do both, using a graph representation the captures the semantics of engineering knowledge. However, this advantage can only be realized if TRAILS is capable of processing various domain-specific meta-models and tool-specific data formats.

Traceability and change management: TRAILS keeps the history and evolution of the artefacts by recording the reasons for changes, the stakeholders involved in realizing a change and all versions of the artefacts. Furthermore, TRAILS does not only track which models have changed but also which of the entities within a model have changed. Therefore, engineers are not only supported in understanding the current structure of the PSS but also they are able to follow its evolution. Consequently, our tool may increase the awareness of stakeholders during development, which leads to a higher acceptance of the design decisions and supports making better decisions in further development.

Although its fundamental technical architecture would allow a further in-depth analysis based on e.g. logical reasoning, TRAILS is however currently dependent on the user to estimate the further consequences of changes in requirements or solution artefacts and to document the reason for a change.

Extendible ontology: Model integration and transformation in order to identify the semantic relationships between entities of heterogeneous models are a central concept of TRAILS. To this end, the integration ontology (see Section 4.1), forms the backbone of model integration by playing the role of a middle language. Moreover, the integration ontology is designed to be adaptable to specific organisational requirements. Therefore, it essentially defines generic types of artefacts, which are common in the development of PSS. Along with this, the ontological entities that these artefacts contain as well as the types of semantic relationships that can exist between them are defined. In addition, hierarchical inheritance structures of artefacts within the ontology ensure the compatibility of TRAILS with many modelling languages. However, at the current stage there are some limitations. To this end, undefined semantic relationships in the integration ontology are resolved through abstraction during the integration process. This means that if a certain type of node or edge is unknown in the integration ontology and no transformation rules have been defined for this situation, the rules for the parent node or edge type are applied. In this case, TRAILS will still be capable of importing the artefact but some of the semantic information that is contained in the original model can be lost, e.g. a very specific type of relationship is replaced with a more generic type.

Visualisation: Another key idea behind TRAILS is to visualise the integrated engineering models in a way that the structure and dynamics of a PSS can be understood intuitively. By presenting the comprehensive engineering knowledge in an appealing and convenient form to the user, they can perform visual analysis of the models making use of the fact that humans are capable of visually detecting complex patterns that machines can't. Accordingly, the main features actualising this goal are: graph layouts and customisable element shapes. Besides, TRAILS provides a matrix presentation, which automatically is derived from the graph-based presentation. In addition, users can create custom views, that highlight some elements while hiding others. However in this context, visual analysis should not been seen as a substitute for rule-based analysis or reasoning but as a complementing instrument.

Simulation: In many cases, there are various alternative PSS designs combining different features of products and services. As every design decision leads to a different performance and costs for the final PSS, it is of high importance to evaluate and prioritise PSS design alternatives (Alfian et al., 2014). However, service components of a PSS impose highly stochastic behaviours to the system, which makes evaluation of a PSS design challenging (Kimita et al., 2012). PSS literature widely proposes simulation as the method to assess PSS designs. To this end, numerous interdependent aspects and measurements need to be addressed in a PSS simulation, such as product and service usage factors e.g. usage frequency and duration, life cycle-related factors, e.g. reusability and maintenance, and environmental impacts (Kimura and Kato, 2002; Garetti et al., 2012). PSS simulation thus needs to consider various of these factors in order to deliver realistic results.

As argued by Zacharewicz et al. (2017) tight model alignment is an essential prerequisite to analyse dynamic dependencies through simulation. In its current version, TRAILS aims at enabling a rather loose coupling between domain-specific modelling avoiding complex and hard to maintain interfaces (or connectors) between different domain-specific modelling tools.

As TRAILS enables integrating different models such as life cycle models as well as structural models, it establishes the basis for such complex simulations. To this point however, the simulation has not been the focus of our work as we primarily aimed at supporting the model-based engineering approach before implementing functions that support model-driven engineering.

However, we believe that TRAILS is perfectly suited to host a simulation engine (e.g. based on system dynamics) that allows analysing dynamic dependencies, such as the impact of resource availability on the result of business processes.

Comparison with other tools: To better clarify TRAILS scope of abilities, it is necessary to compare it with other types of existing tools. Therefore, in the following we compare TRAILS with two major similar types of tools.

There are a plethora of **Requirements Engineering Tools** available such as DOORS[1], Rational Requisite[2], Integrity[3], etc. However, TRAILS does not aim at general management of requirements, but enabling traceability of requirements through models of development artefacts and processes. In addition, as TRAILS was designed to support managing the life cycle of a PSS with a focus on engineering, it provides a high level understanding of how each requirement is satisfied by the PSS.

The other category of tools which TRAILS can be compared, is general **Modelling Tools** like Visual Paradigm[4], Microsoft Visio[5], etc. General modelling tools usually illustrate distinct viewpoints on a specific real world concept, i.e. the component structure of a hardware product or the logical order of activities within a business process. If the intention is to understand a certain part of the PSS from a specific viewpoint, one should use such modelling tools. However, TRAILS takes a different approach by integrating multiple models or perspectives on ths PSS as well as the development process itself. Therefore, TRAILS enables stakeholders to comprehend the interdependencies within the PSS components.

In summary, the current version of TRAILS mainly offers the architectural foundations to implement advanced engineering intelligence features. Today, its core functionality is to import various models specified in DSML and data formats into a comprehensive PSS model that comes as a semantic graph. In doing so, TRAILS relies on the resource description framework as a format for representing the engineering information within a semantic graph. It is therefore of a great importance to draw the limitations of our approach, not only to better reflect the scope of this study, but also to expose limitations which we want to encounter in future work, as we discuss subsequently.

8. Conclusion and future work

PSS are complex socio-technical systems that containing multiple physical, software and service components that need to be seamlessly integrated in order to deliver the desired value-in-use to the customer. Consequentially, the development and life cycle management of PSS demands stakeholders from various engineering domains to work together with each of them using special development methods, modelling languages and engineering software tools. As repeatedly stated in literature, visualisation and analysis of relationships between the different engineering artefacts is essential for stakeholders to understand the interdependencies among components of the system. To close this gap, we presented the our software tool, TRAILS, which enables integrating models from different engineering domains, capturing and visualising the semantic relationships among and within merged engineering artefacts.

To this end, first we focused on developing an integration ontology, which acts as a meta-model to enable model transformation and integration. Besides, the proposed ontology can act as a traceability reference model capturing trace links between various engineering artefacts. However, in order to make the tool more flexible in terms of

[1] http://www-03.ibm.com/software/products/en/ratidoor
[2] https://www-01.ibm.com/software/in/awdtools/reqpro/
[3] http://www.ptc.com/application-lifecycle-management/integrity
[4] https://www.visual-paradigm.com/
[5] https://products.office.com/visio/

compatibility with third party engineering tools a desirable feature for the further development of TRAILS is the possibility to specify model transformation rules through drag and drop combination of atomic transformation operators. This way, domain experts can add new DSMLs or data formats to TRAILS without having to touch the software code by just configuring the transformation process.

In a case study example we presented a comprehensive PSS model that consists of only five sub-models (two of them being modified versions of already existing models). However, in a real industrial case, such a model comprises a much higher number of different sub-models, each of them featuring a higher level of detail. Thus, in a realistic setting, the semantic graph would accumulate to several thousand nodes. As a consequence, TRAILS features like role-based and intuitive filtering need to receive increased efforts. In this context we also see many promising prospects in further developing TRAILS to realise the potentials of visual analytics. If engineers are capable of intuitively comprehending the dependencies within a system, they will most likely be able to provide better solutions.

As next steps, we also will add enhanced team work features to TRAILS. As development of PSS involves high number of people, features to support collaborative work is necessary. Collaborative engineering features are critical for an engineering tool like TRAILS. Currently, the TRAILS back-end database server has limited support for several concurrent users and managing different versions of a PSS structure caused by editing of different users. Regarding better multi-user support, the TRAILS back-end offers various potentials for enhancement including secure user management and data transfer (currently data is served via HTTP), a revision control system, change management and update broadcasts in order to better suit the needs of concurrent engineering.

Since TRAILS uses RDF to represent engineering knowledge within a semantic graph, there is the possibility to enhance the tool by exploiting other standard semantic web technologies, such as SPARQL, OWL or RIF. By doing so, TRAILS can be equipped with advanced semantic search functions. Besides, an inconsistency management mechanism can be developed. For example, based on a user-defined set of rules, inconsistencies among PSS elements, which are imported and merged from domain-specific models, can be detected. Right now there is no automatic inconsistency detection and resolution. Hence, manual inspections are needed to identify conflicts between different models. However, manual inspection is often an exhausting and complicated process suffering from human faults that lead to inaccuracy and incompleteness.

In future, TRAILS should be able to automatically check or support the manual inspection in order to identify the semantic traceability graph inconsistencies. Such inconsistencies could be violation of fundamental physical or logical laws (deadlock in a work-flow because of cyclic control flows; self-containment; negative component weight; dimensions of a component are bigger than its containment; incorrect conversion of measurement units), a mismatch between a requirement and the solution artefact that is supposed to fulfil this requirement, contradicting requirements that refer to the same solution artefacts, an unbound requirement (requirement that is not fulfilled by any solution artefacts), a solution artefact that does not fulfil any requirement (over engineering), a mismatch between the attributes of a subsystem and its components (the aggregate weight of components is higher than the specified weight of the compound). Also, we will implement functionality that aims at using the information captured by TRAILS to simulate service provision in order to optimize the PSS overall architecture.

To sum up, we argued that complex engineering projects, such as the development of PSS, that involve a variety of engineering domains have need for a tool enabling the integration of heterogeneous engineering artefacts into a comprehensive model for purposes like traceability, change management or various analysis tasks. Having introduced our overall concept to tackle this issue, we introduced our prototypical tool TRAILS and presented its core features. We then demonstrated the possible application of TRAILS in an academic PSS engineering project using the example of a bike sharing system. Discussing the current development state, the basic concepts and comparing TRAILS to other types of engineering tools, we concluded that TRAILS is subject to a number of limitations and showed potentials for further enhancing the tool. Nevertheless, in summary the holistic concept of TRAILS, i.e. enabling the integration of various heterogeneous engineering artefacts through abstraction and model transformations provides a fundamental value-added for various stakeholders within the life cycle of a PSS.

Acknowledgements

We thank the German Research Foundation (DFG) for funding this work as part of the collaborative research centre Sonderforschungsbereich 768 - Managing cycles in innovation processes - Integrated development of product-service-systems based on technical products'(SFB768).

References

Alfian, G., Rhee, J., Yoon, B., 2014. A simulation tool for prioritizing product-service system (pss) models in a carsharing service. Comp. Indust. Eng. 70, 59–73.
Baines, T.S., Lightfoot, H.W., Evans, S., Neely, A., Greenough, R., Peppard, J., Roy, R., Shehab, E., Braganza, A., Tiwari, A., et al., 2007. State-of-the-art in product-service systems. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 221 (10), 1543–1552.
Beuren, F.H., Ferreira, M.G.G., Miguel, P.A.C., 2013. Product-service systems: a literature review on integrated products and services. J. Clean. Prod. 47, 222–231.
Bézivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., Lindow, A., 2006. Model Transformations? Transformation Models!. Model driven engineering languages and systems. Springer, pp. 440–453.
Cavalieri, S., Pezzotta, G., 2012. Product–service systems engineering: state of the art and research challenges. Comput. Ind. 63 (4), 278–288.
Chen, D., Doumeingts, G., Vernadat, F., 2008. Architectures for enterprise integration and interoperability: past, present and future. Comput. Ind. 59 (7), 647–659.
Czarnecki, K., Helsen, S., 2006. Feature-based survey of model transformation approaches. IBM Syst. J. 45 (3), 621–645.
Dorfman, M., 1984. Arts an automated requirements traceability system. Journal of Systems and Software 4 (1), 63–74.
Eisenbart, B., Blessing, L., Gericke, K., et al., 2012. Functional modelling perspectives across disciplines: a literature review. Proceedings of 12th international design conference.
Frank, M., Harel, A., Orion, U., 2014. Choosing the appropriate integration approach in systems projects. Systems Engineering 17 (2), 213–224.
Garetti, M., Rosa, P., Terzi, S., 2012. Life cycle simulation for the design of product-service systems. Comput. Ind. 63 (4), 361–369.
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.
Geum, Y., Park, Y., 2011. Designing the sustainable product-service integration: a product-service blueprint approach. J. Clean. Prod. 19 (14), 1601–1614.
Hajimohammadi, A., Cavalcante, J., Gzara, L., 2017. Ontology for the pss lifecycle management. Procedia CIRP 64, 151–156.
Kernschmidt, K., Wolfenstetter, T., Münzberg, C., Kammerl, D., Goswami, S., Lindemann, U., Krcmar, H., Vogel-Heuser, B., 2013. Concept for an integration-framework to enable the crossdisciplinary development of product-service systems. IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) 2013. IEEE, pp. 340–345.
Kimita, K., Tateyama, T., Shimomura, Y., 2012. Process simulation method for product-service systems design. Procedia CIRP 3, 489–494.
Kimura, F., Kato, S., 2002. Life cycle management for improving product service quality. Proceedings of the 9th International Seminar on Life Cycle Engineering, Erlangen (Germany). pp. 25–31.
Klingner, S., Becker, M., 2015. Formal modelling of components and dependencies for configuring product-service-systems. Enterprise Modelling and Information Systems Architectures 7 (1), 44–66.
Komoto, H., Tomiyama, T., 2008. Integration of a service cad and a life cycle simulator. CIRP Annals-Manufacturing Technology 57 (1), 9–12.
Lee, S., Kim, Y., 2010. A product-service systems design method integrating service function and service activity and case studies. Proceedings of the 2nd CIRP international conference on industrial product service systems. pp. 275–282.
Lim, C.-H., Kim, K.-J., Hong, Y.-S., Park, K., 2012. Pss board: a structured tool for product–service system process visualization. J. Clean. Prod. 37, 42–53.
Lynn Shostack, G., 1982. How to design a service. Eur. J. Mark. 16 (1), 49–63.
Mäder, P., Orlena, G., 2011. Towards automated traceability maintenance. J. Syst. Softw. 85 (10), 2205–2227.
Marques, P., Cunha, P.F., Valente, F., Leitão, A., 2013. A methodology for product-service systems development. Procedia CIRP 7, 371–376.
Maussang, N., Zwolinski, P., Brissaud, D., 2009. Product-service system design methodology: from the pss architecture design to the products specifications. J. Eng. Des. 20 (4), 349–366.

Medvidovic, N., Grünbacher, P., Egyed, A., Boehm, B., 2003. Bridging models across the software lifecycle. J. Syst. Softw. 68 (3), 199–215.
Meier, H., Roy, R., Seliger, G., 2010. Industrial product-service systems ips 2. CIRP Annals-Manufacturing Technology 59 (2), 607–627.
Mens, T., Van Gorp, P., 2006. A taxonomy of model transformation. Electron Notes Theor Comput Sci 152, 125–142.
Morelli, N., 2006. Developing new product service systems (pss): methodologies and operational tools. J. Clean. Prod. 14 (17), 1495–1501.
Nemoto, Y., Akasaka, F., Shimomura, Y., 2015. A framework for managing and utilizing product-service system design knowledge. Production Planning & Control 26 (14–15), 1278–1289.
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. Comput. Ind. 77, 1–14.
Sakao, T., Shimomura, Y., Sundin, E., Comstock, M., 2009. Modeling design objects in cad system for service/product engineering. Comput.-Aided Des. 41 (3), 197–213.
Song, W., 2017. Requirement management for product-service systems: status review and future trends. Comput. Ind. 85, 11–22.
Tang, A., Jin, Y., Han, J., 2007. A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software 80 (6), 918–934.
Tilstra, A., Campbell, M., Wood, K., Seepersad, C., 2010. Comparing matrix-based and graph-based representations for product design. DSM 2010: Proceedings of the 12th International DSM Conference. The Design Society, pp. 195–199.
Van Halen, C., Vezzoli, C., Wimmer, R., 2005. Methodology for product service system innovation: how to develop clean, clever and competitive strategies in companies. Uitgeverij Van Gorcum.
Varró, D., Pataricza, A., 2004. Generic and Meta-transformations for Model Transformation Engineering. UML 2004 The Unified Modeling Language. Modeling Languages and Applications. Springer, pp. 290–304.
Vasantha, G., Roy, R., Lelah, A., Brissaud, D., 2012. A review of product-service systems design methodologies. J. Eng. Des. 23 (9), 635–659.
Vilela, J., Castro, J., Martins, L.E.G., Gorschek, T., 2017. Integration between requirements engineering and safety analysis: a systematic literature review. Journal of Systems and Software 125, 68–92.
Wiesner, S., Freitag, M., Westphal, I., Thoben, K.-D., 2015. Interactions between service and product lifecycle management. Procedia CIRP 30, 36–41.
Wolfenstetter, T., Füller, K., Böhm, M., Krcmar, H., Bründl, S., 2015. Towards a requirements traceability reference model for product service systems. International Conference on Industrial Engineering and Systems Management (IESM) 2015. IEEE, pp. 1213–1220.
Wolfenstetter, T., Kernschmidt, K., Münzberg, C., Kammerl, D., Goswami, S., Lindemann, U., Vogel-Heuser, B., Krcmar, H., 2014. Supporting the cross-disciplinary development of product-service systems through model transformations. IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) 2014. IEEE, pp. 174–178.
Zacharewicz, G., Diallo, S., Ducq, Y., Agostinho, C., Jardim-Goncalves, R., Bazoun, H., Wang, Z., Doumeingts, G., 2017. Model-based approaches for interoperability of next generation enterprise information systems: state of the art and future challenges. Information Systems and e-Business Management 15 (2), 229–256.

Thomas Wolfenstetter is a PhD student at the Chair for Information Systems at Technische Universität München (TUM), Germany. He studied information Systems at TUM and at the School of Economics and Management at Tongji University, Shanghai, PR China. During his studies he has been a working student for BMW Group, product data management strategy department. He is a member of the collaborative research centre SFB768 Managing Cycles in Innovation Processes where he researches requirements traceability for product service systems. His research interests lie in the areas of service engineering, product service systems, requirements engineering and management of innovation processes.

Mohammad Basirati is a PhD student and research assistant at the Chair for Information Systems, Technische Universität München (TUM). He graduated in informatics from TUM in 2016. His main research interests lie in requirements engineering, product service systems and model-based systems engineering. Within the collaborative research centre SFB768 Managing Cycles in Innovation Processes he works on innovative software tools that are capable of automatically identifying inconsistencies between engineering artefacts in the context of in model-based systems engineering.

Markus Böhm is a research group leader at the Chair for Information Systems at Technische Universität München, Germany. He regularly publishes research in the areas of corporate transactions, digital business models and innovation potentials of novel technologies. Among his core areas of research are the role of IT in the context of mergers & acquisitions, digital business models and the innovation potentials of novel technologies for the transformation of existing and development of new business models.

Helmut Krcmar is a full professor of Information Systems and holds the Chair for Information Systems at the Department of Informatics, Technische Universität München, Germany, since 2002. He worked as a postdoctoral fellow at the IBM Los Angeles Scientific Centre, as assistant professor of Information Systems at the Leonard Stern School of Business, New York University, and at Baruch College, City University of New York. From 1987 to 2002 he was Chair for Information Systems, Hohenheim University, Stuttgart. His research interests include information and knowledge management, IT-enabled value webs, service management, information systems in health care and e-government.

Not Included in Review and Evaluation: P6

Facilitating Consistency of Business Model and Technical Models in Product-Service-Systems Development: An Ontology Approach

M. Zou*, M. R. Basirati, H. Bauer***, N. Kattner****, G. Reinhart***, U. Lindemann****, M. Böhm**, H. Krcmar** and B. Vogel-Heuser***

**Institute of Automation and Information Systems, Technical University of Munich, Garching, Germany (e-mail: minjie.zou@tum.de, vogel-heuser@tum.de)*
***Chair for Information Systems, Technical University of Munich, Garching, Germany (e-mail: mohammadreza.Basirati@in.tum.de, markus.boehm@in.tum.de, krcmar@in.tum.de)*
****Institute for Machine Tools and Industrial Management, Technical University of Munich, Garching, Germany (e-mail: harald.bauer@iwb.mw.tum.de, gunther.reinhart@iwb.mw.tum.de)*
*****Institute of Product Development, Technical University of Munich, Garching, Germany (e-mail: niklas.kattner@tum.de, udo.lindemann@tum.de)*

Abstract: Due to the fast growing and ever changing innovation cycles, industries are changing their strategies from a product-centric to service-centric approach, leading to the emergence of product-service systems (PSS) which integrate services into physical technical systems. In the model-based development of PSS, various models are employed by different stakeholders to represent their views on the system. However, there is a high diversity of these models in both forms and contents. Among others, business models and technical models come in different levels of abstraction, and thus, are hard to align with each other. In this study, we propose an approach to support PSS development by relating business models to technical models using ontology. Web ontology language (OWL) is employed to describe PSS knowledge, the Query Language SPARQL and Semantic Web Rule Language (SWRL) are used for consistency checking and reasoning potential inconsistencies. © 2019 IFAC

Keywords: product-service systems (PSS), model-based engineering, ontology, aligning business and technical models, model consistency.

1. INTRODUCTION

The competitive global economy has forced industries to shift from a product-centric approach to a more service-centric approach (Meier, Roy and Seliger, 2010). With this regard, product-service system (PSS) as a new type of systems is introduced. PSS incorporates both physical products and services into a single working system. Through offering services, PSS establishes a stronger relationship with end-users. Moreover, it lowers the dependency of manufacturer on environmental resources as part of value proposition is made by services (Baines et al., 2013).

Nevertheless, simultaneous design and development of services and physical products, which are from different nature, introduces new challenges. For example, PSS development requires intense collaboration among business and technical teams through the whole lifecycle in order to tackle alignment of services and products (Wolfenstetter et al., 2018). However, in practice, different components of a system are mostly developed in separate teams, while every team uses a different modeling language based on its domain (Song, 2017).

In a model-based development, developers use various types of models, from prescriptive and formal models, to descriptive and informal models. The advantage of model-based development is that each model can focus on the key problem from a particular view and restrict the degrees of freedom. However, different models have varying forms and languages, resulting in great difficulty in cooperation across models.

One important aspect of PSS is how business models and decisions influence technical structures, and vice versa. In the context of business intelligence, systems usually combine data gathering, data storage, and knowledge management with analytical tools to present complex information to decision makers of the company. Therefore, combining business and technical concerns from their models is vital in PSS.

In this study, we mainly focus on correlating business and technical models in PSS. Two questions will be addressed: how to integrate the heterogeneous models created during the PSS innovation process, and how to assist different stakeholders to identify inconsistencies and make decisions.

This paper addresses the above-mentioned problem by introducing an ontology-driven approach. To this end, we establish an ontology, which represents the models, the models' attributes and the relationship among them. We selected models that are relevant for business design, product design, software design, and manufacturing. Based on such an ontology, we show how searches and reasoning can be employed to address connection of the different models.

2. STATE OF THE ART

The goal of this study is to have a method to correlate the information from business and technical models, and to ensure decisions in business models will not lead to conflicts i.e. inconsistencies in technical designs. Correspondingly, following requirements should be met:

a) A common representation form for abstract business models and concrete technical models;

b) A common understanding i.e. vocabulary interoperability between technical and business models;

c) Consistency check between business and technical development models;

d) Tool assistance for managers and engineers.

Based on these requirements, the state of the art will be introduced and analyzed.

2.1 Modeling and Analysis Methods for PSS

A comprehensive holistic system model can be adopted to integrate domain-specific models (Lukei et al., 2016), which serves as a general description of the system including the requirements, the structure and the behavior. For example, Thramboulidis (2013) used the Systems Modeling Language (SysML), a graphical modeling language to integrate models in mechatronic systems. But business strategies need to be modelled separately, e.g. with the Business Process Modeling Notation (BPMN) or models describing the exchange among participating actors, such as the e3value model (Gordijn and Akkermans, 2001).

Similarly, Gausemeier et al. (2009) aimed to synchronize changes between domain-spanning and domain-specific models in the development of complex mechatronic systems in order to keep information consistent. In addition, Kaiser (2013) made contributions to a holistic and general interdisciplinary description for systems modeling. However, their approach is limited to a technique, in which graphical modeling tools (triple graph) and a model transforming and updating engine are required.

Broy and Reussner (2010) defined a single underlying model (SUM) and a single underlying meta-model that captures the information portrayed in all views on a system. This approach can ensure information consistency among models of requirements, architecture, and code (Konersmann et al., 2013). However, it only addressed models in the software development.

Maisenbacher et al. (2014) has applied agent-based modeling to support PSS development, in which a system is modeled as a collection of autonomous entities (agents). However, neither a specific technical model nor a business model was investigated.

Weidmann et at. (2015) employed discrete event simulation models, i.e. a network of queues and activities, to model product related services in product oriented PSS. But it doesn't support representing concrete technical models. Furthermore, due to its low formality, consistency check and tool assistance are not possible.

Ontologies can also be used in the interdisciplinary PSS development, which is commonly defined as an explicit formal specification of the terms in the domain and the relations among them (Gruber, 1993). Common understanding of terms within the domain could be an important measure to define formal specification. An engineering knowledge base (EKB) was proposed on top of a common data repository to store explicit engineering knowledge (Biffl et al., 2009; Moser and Biffl, 2012). Malcki and Belkadi (2018) introduced the PSS ontology to industrial PSSs, in which domain knowledge regarding the product component, service, sensor, measurement, and connection constraints were stored and linked to support collaborative development. Similarly, Correia and Stokic (2017) developed a user-centric PSS ontology to describe concepts relevant for the dynamics of the context, and used it to improve the communication among various stakeholders. Wang and Ming (2014) proposed a general modularity method of product-service to provide a customization-oriented menu based on the ontology knowledge.

Though ontology-based approaches can provide a common understanding and data repository among models, the consistency checking still requires extra techniques.

2.2 Facilitating consistency in PSS

Song and Sakao (2016) investigated conflicts among services in product-service offerings by employing linguistic techniques, while Shimomura and Hara (2010) introduced a conflict resolution method for PSS development. Conflicts include the inconsistent names, descriptions of functions, variables and values. Wimmer et al. (2018) proposed to adopt AutomationML, a standard unifying data exchange format, to represent and connect the different engineering models. He also developed a dedicated query language to check consistency in this aggregated data file. Egyed et al. (2018) developed an incremental approach to detect inconsistencies among artifacts in software engineering. Nevertheless, these studies did not address how heterogeneity of PSS models can be handled.

Feldmann et al. (2015) and Zou et al. (2018) worked on the industrial PSS and inconsistencies among interdisciplinary designs for technical products. Despite a similar motivation with this study, they didn't include business strategies in PSS. Basirati et al. (2018) proposed a method to tackle inconsistencies in and between the service side and the product side. But it was limited to data inconsistencies among technical models.

Overall, despite the multitude of approaches within model-based development of PSS, our requirements cannot be fully met by existing work and need more research efforts.

3. CONSTRUCTING A PSS ONTOLOGY UTILIZING MODEL-BASED DEVELOPMENT

In this section, a set of models from different domains are introduced, which are frequently used in PSS development. Fig. 1 shows the overall structure of how the information from different models is collected, processed and delivered to the stakeholders. Information is captured and presented in concrete models (left side of the Fig. 1) and later transformed to high level managerial roles for decision makings.

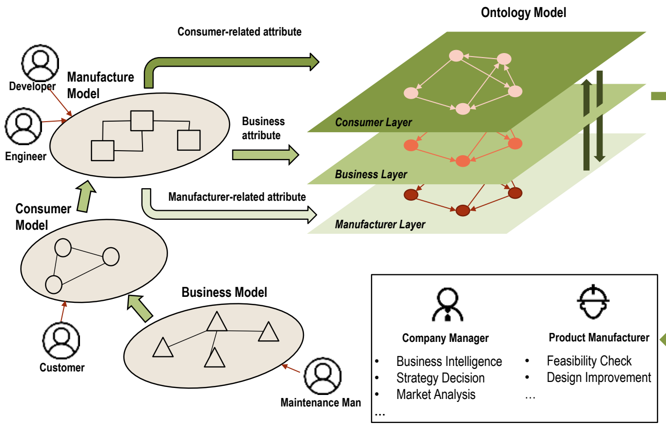


Fig. 1. Architecture of the PSS ontology representing interdisciplinary developing models

Business Canvas Model (BCM) is a template that captures business characteristics into nine different topics such as customer segments, key partners, value propositions, cost structure and so on. BCM is the most abstract model, which is only used by the board management.

Another high level model is Consumer Decision Process (CDP) model. In a PSS, we have a longer and more intense relationship with customers, it is of high importance to model their decision making process in a model. To this end, CDP describes the decision-making process of a consumer and contains several stages, such as need recognition, information research, alternative evaluation, and purchase and after purchase evaluation. The consumer model can connect the manufacture side and the business side, since consumers pay the company for the product made by the manufacturer and it is regarded as a key model.

SysML*4Mechatronics* is a technical model based on SysML that contains most relevant information of a physical part from the perspective of mechatronics implementation.

An important part of any PSS is software, which connects services to the products. To design and implement software many models are used. Nevertheless, one of the most important and frequently used models is UML sequence diagram model. This model captures the sequences of important activities and processes –mostly triggered by the user - that the software must be able to handle. As in this paper we only refer to this type of UML model, in the rest of this paper, whenever we mention UML, we mean UML sequence diagram model.

Finally, the manufacturing model contains the information of product design and manufacturing process, including both physical and software products.

When representing the relations among models in the ontology, defining all relations simultaneously is often difficult. Therefore, a classification of these relationships is helpful. To transform the models into an ontology, in our approach, we divide the models and their attributes into three layers, namely, consumer level, business layer and manufacturing layer (depicted in Figure 1, right side).

This classification is based on the stakeholders in PSS, i.e. the manager, the manufacturer (including engineers and service developers) and the consumer. They have different targets and constraints, while they interact and are constrained by each other (Fig. 2). The manager aims to maximize the profit and is constrained by consumer demand. Target of the consumer is high value of the provided PSS value, which is limited by the price and their income. Manufacturers and engineers are responsible to develop a feasible PSS under limitations of the budget, technologies etc. Interactions and relations between these roles can be finance-related (i.e. revenue, price, cost), document-related (i.e. requirements, demand expression) as well as role-related (i.e. candidate value supplier).

Based on the roles, defined layers and relationships (depicted in Figure 2), we establish the ontology. In first step, we formulate the concepts of each model into the ontology. For example, for BCM, we have concepts such as customer segments and key partners or for SysML*4Mechatronics* we will have module, software block, mechanical block and so on. Afterwards, we put each attribute of the model in a specific layer that it belongs to. Finally, we incorporate the relationships among the models and their concepts (shown in Figure 2) into the ontology.

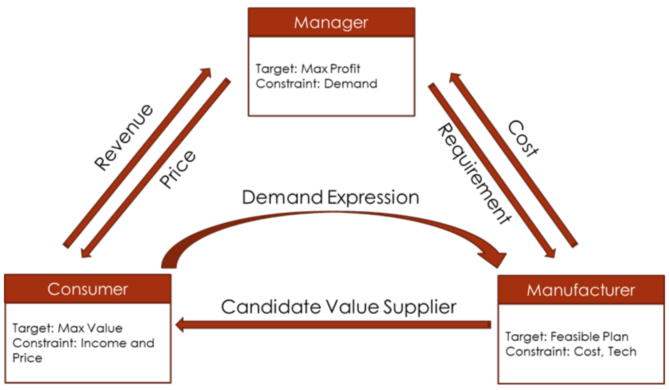


Fig. 2. Interaction between stakeholders

4. CONSISTENCY IN PSS DEVELOPMENT

Maintaining consistency among all different aspects of any system is challenging. However, even more complex is to check the dependencies of business and service side of a PSS with its physical side. At the same time, offering services upon a physical layer intensifies inter-connection between business and technical components during PSS development.

We employ the ontology approach of this study to tackle this problem (Moser and Biffl, 2009).

First, the use cases are classified in which the ontology can support decision making and maintaining consistency among different components, during PSS development. Three major use cases are identified as follows:

1) **Service Innovation**: The core added-value of a PSS originates in its ability to offer new services. Hence, the ability to analyze a new service feature and its impact on other components, as well as the cost and the revenue, is significant.

2) **Partnership and Market Selection**: PSS development necessitates a high number of partnerships with suppliers. In most cases, a single firm has limited capacity to produce all parts and provide all services in-house. Therefore, a critical issue is how the outcome of a PSS will be affected if there is a change in the partners. In addition, PSS increases the relationship between a firm and the end-users. Therefore, knowing which users PSS should establish a relationship with or in which areas the services should be provided, is vital for the design and development.

3) **Error and Inconsistency Detection**: PSSs are complex systems with heterogeneous components and aspects caused by integrating products and services. Therefore, tracing a single defect and its impact on the overall system is challenging. By applying ontologies and implicit meanings of data, the dependencies become more explicit and automatic methods can be used to reason (Apostolski et al., 2010).

Afterwards, several methods are applied for these purposes: to query information and check consistency using SPARQL queries (Word Wide Web Consortium, 2013), and to derive relations and conduct reasoning using SWRL rules (Word Wide Web Consortium, 2004). Applying SPARQL queries on the ontology, would enable us to identify the dependencies among the components. Consequently, we would be able to incorporate such dependencies and their influences in the decision makings for the use cases. Moreover, by assigning quantifiable values to the attributes defined in the ontology such as cost, price or component-specific properties, we can calculate a preliminary estimation quickly and easily with SWRL. In an exemplary case study, we elaborate how the ontology approach can be applied in PSS development.

Technically, SPARQL is like the well-known structured Query Language (SQL) supported by most relational database systems. SPARQL queries consist of two parts: a clause identifying the type of query (and when retrieving information, a list of information to be returned), and a pattern to be matched against the RDF data (Word Wide Web Consortium, 2013).

By contrast, SWRL is adopted mainly due to its power in deduction reasoning. Typical examples are mathematical text problems and syllogistic inferences.

5. CASE STUDY: AN E-BIKE SHARING PSS

In this section, the proposed method and PSS ontology is applied on an e-bike sharing system as an example of PSS. The e-bike-sharing system, called PSSycle, is developed by graduate students in the context of an interdisciplinary research project. In the following section, the modeling process and an inconsistency between the business strategy and the engineering process are first described, and afterwards, the proposed ontology is applied to detect the inconsistency.

5.1 Case Description

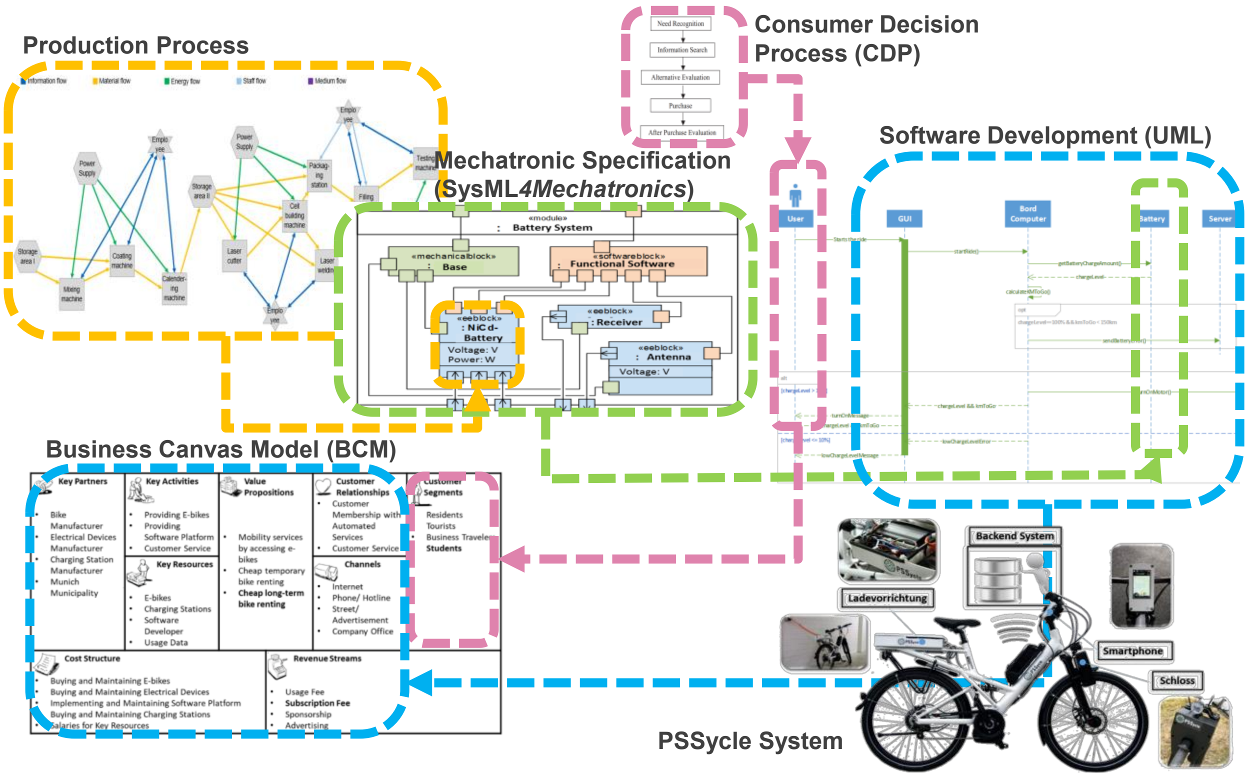
PSSycle is an e-bike sharing system that offers e-bike rental and its related services. To borrow a bike, users can search for nearby e-bikes and book one via the app or directly using the on-board computer of the e-bikes. Customers can end the rental either roadside or at charging stations. The battery is charged at these charging stations. To develop such a PSS, a wide variety of models are used. For this study, five models from different domains and levels are analyzed (Fig. 3). Two models, namely, SysML*4Mechatronics* and Production Process, represent the battery specifications and manufacturing respectively. The overall business model is captured in BCM. Moreover, CDP model elaborates the user perspective, while a software development model in form of the UML Sequence Diagram presents how the app and other components work together to unlock an e-bike.

The pink line shows how consumers of CDP and BCM as well as the engineers who developed the UML model, refer to the same entity and connect these models. The blue line shows the connection among components that are mentioned in both the UML and BCM models. The green line displays how the battery in the UML model and its specification in SysML*4Mechatronic* are connected. Finally, the orange line connects the battery specification and its manufacturing process. These connections are a few examples among a high number of potential dependencies among the models.

The information in these models can be represented in a shared ontology, so that the demand of consumers can be more efficiently delivered to managers and manufacturers. Therefore, the company can offer a suitable product for the market more promptly. We divide all attributes into three layers: business layer, physical layer and consumer layer. The business layer contains the attributes related to revenue and cost, with which the manager can easily understand the change of the net profit. The physical layer includes all the technical specifications of the e-bike and its components such as the battery and motor. Consumer layer includes the attributes that represent consumers' rationale and are relevant to the decision process model.

Protégé[1] is used to create ontologies of the described models and their relationships. Fig. 4 shows the E-bike sharing ontology to couple the business and technical models.

[1] https://protege.stanford.edu/



Production Process
Consumer Decision Process (CDP)
Software Development (UML)
Mechatronic Specification (SysML4Mechatronics)
Business Canvas Model (BCM)
PSSycle System
Ladevorrichtung
Backend System
Smartphone
Schloss

Fig. 3. Models used in PSSycle as an example of PSS engineering

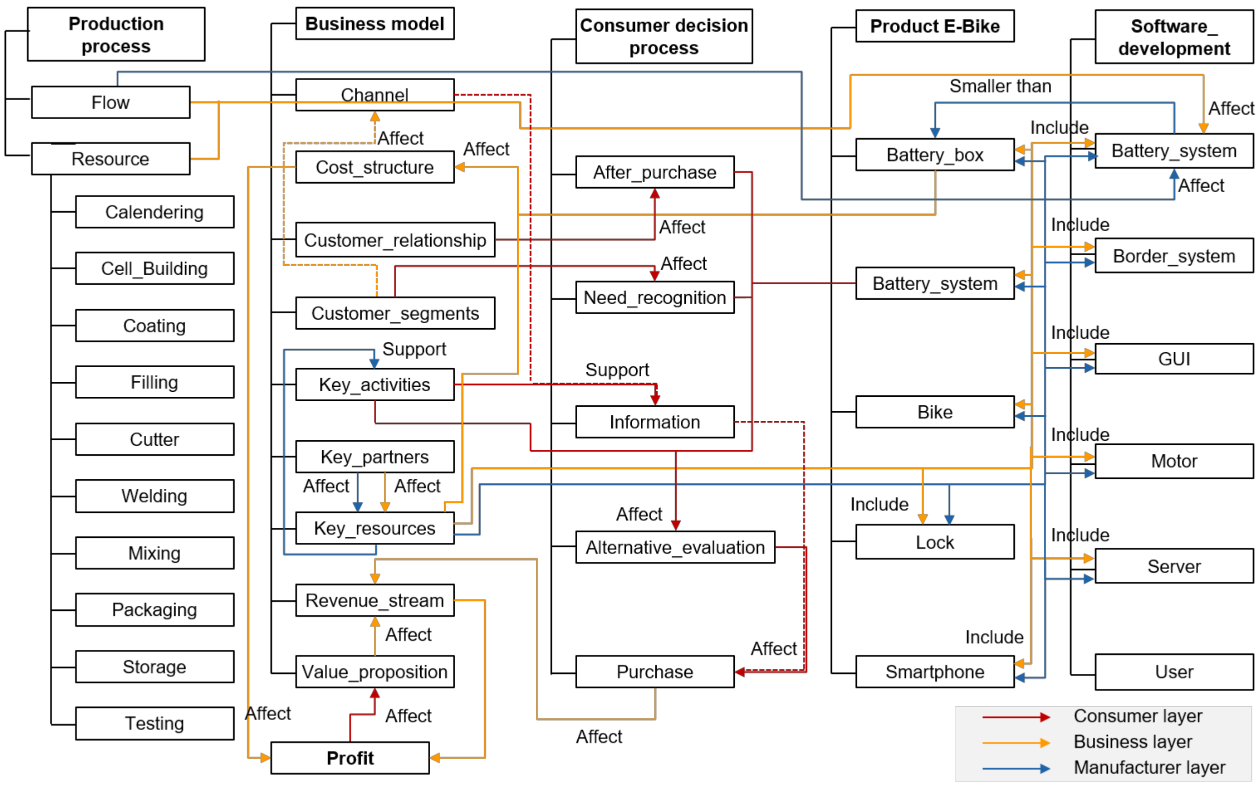


Fig. 4. PSS ontology for PSSycle

The information in the shared ontology can be further processed, and the key information will be delivered to managers and manufacturers in an organized way.

5.2 Reasoning: Selecting Battery Supplier

To select the battery manufacturer partner, we analyze the impact of the manufacturer on the PSSycle using the ontology. In the first step, we find all battery related elements among the models. The battery manufacturer is listed in the key partners of the business model canvas. The battery, including its physical specifications is modelled in SysML*4Mechatronics*. Afterwards, the battery is mentioned in the manufacturing process model to present the step for assembling the battery and the battery box. In the next step, we evaluate the impact of these elements on the three layers of business, physical and consumer. On the physical layer, change of the battery affects the battery box and consequently the manufacturing machinery, which are used for assembling a new battery and its box. This indirectly affects the cost on the business layer. In addition, the battery is connected to the board-computer and the software part modelled in UML sequence diagram. On the business layer, in addition to the potential costs for the manufacturing process, the price of the battery itself is affected based on which type of battery is used. On the consumer layer, the decision process is affected by the battery capacity (what range of riding the battery will support) and indirectly by the use price in case the battery price will change it.

To analyze such a scenario, we employ SPARQL to search the attributes for every layer and find their value. As an example, the following query can be used to check factors affecting customers' evaluation ranking as well as factors affecting customers' purchase decision directly:

```
SELECT ?n ?m
WHERE {          ?n :support Evaluation_Ranking.
                 ?m rdf:type :Purchase.
                 ?m : affect Business_Layer_Purchase. }
```

After collecting the values, based on predefined rules, impacted attributes can be calculated. For every manufacturer we would be able to calculate the costs, price, and accordingly its impact on consumer decision process. For instance, price of the charging station can be calculated by SWRL as it follows:

```
Number_of_Charging_Station(?number) ^…
      Single_Price_of_Charging_Station(?price) ^ …
         swrlb:multiply(?purchase, ?number, ?price) ->…
                Purchase_of_Charging_Station(?purchase)
```

SWRL can also be used for conditional judgement. For example, "only when the battery box size is larger than the battery size, the battery system feasibility is true. Only in this case, the battery box can contain the battery" can be formulated as:

```
Battery_Box_Size(?box) ^ Battery_Size(?battery) ^ …
     swrlb:greaterThan(?box,?battery) ->…
         Battery_System_Feasibility(1, ?box, ?battery)
```

Therefore, establishing the ontology, particularly using the process described in Section 3, enables us to find concepts, attributes, values and the relationships among heterogeneous models used in PSS development. Moreover, we are able to perform some basic calculations and consistency checks among the models. Consequently, during design and development of a PSS, the stakeholders in different levels and domains would be able to understand the influences of their relevant models on the PSS as a whole. In the example of this case study, the managerial board could better evaluate impact of each battery supplier and their batteries on the products and service offerings of the PSSycle.

6. CONCLUSION AND FUTURE WORK

In this study, an ontology-based approach is demonstrated to couple the interdisciplinary models representing views of different stakeholders in PSS: business strategy makers, manufacturers, and consumers. The proposed approach allows a semi-automatic consistency checking between business and manufacturing models, provides reasoning and avoiding potential inconsistencies, as well as assists in selecting partners, markets and technologies. Through the proposed approach, the manufacturer and manager are enabled to have quicker reactions to the change of the market, and better meet customers' demands. The developed ontology is applied and evaluated in a demonstrative PSS (PSSycle), but its applicability still needs further investigation when the system complexity scales up. In addition, the models of different abstraction levels may result in ontology maintenance problems. This is challenging because changes in business models are usually minor whereas manufacturing models vary a lot, which are caused by the minor change.

Nevertheless, there remains room for improvement. For potential future work, the ontologies can be combined with simulation methods to check the dynamical interactions between disciplines. Furthermore, a systematic reasoning mechanism of the PSS ontology will be developed to realize more intelligent automatic analysis in PSS.

ACKNOWLEDGMENT

We acknowledge the German Research Foundation (DFG) for funding this work as part of the research in the Collaborative Research Centre SFB 768: Managing cycles in innovation processes – integrated development of product-service systems based on technical products (SFB 768/3, Subproject D1) at the Technical University of Munich.

REFERENCES

Apostolski, V., Stojanov, R., Jovanovik, M., and Trajanov, D. (2010): *Use of Semantic Web Technologies in Knowledge Management*, Proceedings of the 7th Conference On Informatics And Information Technology (CIIT), pp. 91 - 96.

Baines, T. S., Lightfoot, H.W., Evans, S., Neely, A., and Greenough, R. (2007): *State-of-the-art in product service systems*, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 221, no. 10, pp. 1543–1552.

Basirati M. R., Zou M., Bauer H., Kattner N., Reinhart G., Lindemann U., Böhm M., Krcmar H. and Vogel-Heuser

B., (2018): *Towards Systematic Inconsistency Identification For Product Service Systems*, Proceedings of the DESIGN 2018 15th International Design Conference.

Biffl S., Schatten A. and Zoitl A., (2009): *Integration of heterogeneous engineering environments for the automation systems lifecycle*, 7th IEEE International Conference on Industrial Informatics.

Broy M. and Reussner R., (2010): *Architectural Concepts in Programming Languages*, Computer, vol. 43, no. 10, pp. 88–91.

Correia A., Stokic D., Siafaka R. and Scholze S., (2017): *Ontology for colaborative development of product service systems based on basic formal ontology*, 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC).

Egyed A., Zeman K., Hehenberger P. and Demuth A., (2018): *Maintaining Consistency across Engineering Artifacts*, Computer, vol. 51, no. 2, pp. 28–35.

Feldmann S., Herzig S. J., Kernschmidt K., Wolfenstetter T., Kammerl D., Qamar A., Lindemann U., Krcmar H., Paredis C. J. and Vogel-Heuser B., (2015): *Towards Effective Management of Inconsistencies in Model-Based Engineering of Automated Production System*s, IFAC-PapersOnLine, vol. 48, no. 3, pp. 916–923.

Gausemeier J., Schäfer W., Greenyer J., Kahl S., Pook S. and Rieke J., (2009): *Management of cross-domain model consistency during the development of advanced mechatronic systems*, Proceedings of the 17th International Conference on Engineering Design, vol. 6, Design Methods and Tools, USA.

Gordijn J. and Akkermans H., (2001): *Designing and evaluating e-business models*, IEEE Intelligent Systems, vol. 16, no. 4, pp. 11–17.

Gruber T. R., (1993): *A translation approach to portable ontology specifications*, Knowledge Acquisition, vol. 5, no. 2, pp. 199–220.

Kaiser L., (2013): *Rahnmenwerk zur Modellierung einer plausiblen Systemstruktur mechatronischer Systeme*, dissertation.

Konersmann M., Durdik Z., Goedicke M., and Reussner R. H., (2013): *Towards architecture-centric evolution of long-living systems (the ADVERT approach),* Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures - QoSA 13.

Lukei, M., Hassan, B., Dumitrescu, R., Sigges, T., and Derksen, V., (2016): *Requirement analysis of inspection equipment for integrative mechatronic product and production system development: Model-based systems engineering approach*, Annual IEEE Systems Conference (SysCon).

Maisenbacher S., Weidmann D., Kasperek D., and Omer M., (2014): *Applicability of Agent-based Modeling for Supporting Product-service System Development*, Procedia CIRP, vol. 16, pp. 356–361.

Maleki E., Belkadi F., Boli N., Zwaag B. J. V. D., Alexopoulos K., Koukas S., Marin-Perianu M., Bernard A. and Mourtzis D., (2018): *Ontology-Based Framework Enabling Smart Product-Service Systems: Application of Sensing Systems for Machine Health Monitoring*, IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4496–4505.

Meier, H., Roy, R., and Seliger, G., (2010): *Industrial Product-Service Systems—IPS 2*, CIRP Annals, vol. 59, no. 2, pp. 607–627.

Moser T. and Biffl S., (2012): *Semantic Integration of Software and Systems Engineering Environments*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 1, pp. 38–50.

Shimomura Y. and Hara T., (2010): *Method for supporting conflict resolution for efficient PSS development,* CIRP Annals, vol. 59, no. 1, pp. 191–194.

Song, W., (2017): *Requirement management for product-service systems: Status review and future trends*, Computers in Industry, vol. 85, pp. 11–22.

Song W. and Sakao T., (2016): *Service conflict identification and resolution for design of product–service offerings*, Computers & Industrial Engineering, vol. 98, pp. 91–101.

Thramboulidis, K., (2013): *Overcoming Mechatronic Design Challenges: the 3+1 SysML-view Model*, Computing Science and Technology International Journal, vol. 1, no. 1, pp. 6–14.

Wang P., Ming X., Wu Z., Zheng M. and Xu Z., (2014): *Research on industrial product–service configuration driven by value demands based on ontology modeling*, Computers in Industry, vol. 65, no. 2, pp. 247–257.

Weidmann D., Maisenbacher S., Kasperek D. and Maurer M., (2015): *Product-Service System development with Discrete Event Simulation modeling dynamic behavior in Product-Service Systems*, Annual IEEE Systems Conference (SysCon) Proceedings.

Wimmer M. and Mazak A., (2018): *From AutomationML to AutomationQL: A By-Example Query Language for CPPS Engineering Models*, IEEE 14th International Conference on Automation Science and Engineering (CASE).

Wolfenstetter, T., Basirati, M. R., Böhm, M., and 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, vol. 144, pp. 342–355.

World Wide Web Consortium W3C (2013): *SPARQL Protocol and RDF Query Language 1.1*, Same Origin Policy - Web Security, [Online] http://www.w3.org/TR/sparql11-overview/ [04.12.2018].

World Wide Web Consortium W3C (2004): *SWRL: A Semantic Web Rule Language*, Same Origin Policy - Web Security. [Online] https://www.w3.org/Submission/SWRL/. [04.12.2018].

Zou M., Lu B. and Vogel-Heuser B., (2018): *Resolving Inconsistencies Optimally in the Model-Based Development of Production Systems,* IEEE 14th International Conference on Automation Science and Engineering (CASE).