Facilitating PSS Development Collaborations by Identifying Conflict Factors and Requirements Dependencies

Mohammad R. Basirati, Chair for Information Systems, Technical University of Munich (basirati@in.tum.de)
Sebastian Hermes, Chair for Information Systems, Technical University of Munich (sebastian.hermes@in.tum.de)

Introduction

We measure success of every system by the degree that it can fulfil its requirements (Nuseibeh and Easterbrook 2000). Product-service systems (PSS) - as an integrated system of physical products and services - particularly have a large set of requirements, which are derived from different worlds of service, software and physical products (Berkovich et al. 2014). For instance, a car-sharing system is a PSS, which integrates car as a product with sharing services. Development of such a system brings together teams such as service design and smartphone app development with traditional mechanics engineering teams. Such different teams have to collaborate intensively and continuously in order to elicit and manage requirements. With this regard, two issues arise. First, collaboration in such a heterogeneous setting of PSS development leads to a higher number of conflicts. Second, we need to identify how the requirements relate to each other not only to manage the requirements better, but also in order to form corresponding collaborations for realizing them. Nevertheless, different teams determine the requirements separately and the knowledge of requirements interdependencies is missing. Such a knowledge is crucial in collaborations and activities with regard to change management, requirements prioritization and testing.

To facilitate efficient collaborations during PSS development, subproject A4 of CRC 768 looked at which factors raise conflicts during requirements management collaborations; and how the relationships among the requirements, written in natural language, can be identified automatically.

CONFLICTS DURING REQUIREMENTS MANAGEMENT COLLABORATIONS

Development of PSS necessitates continuous collaboration of different teams such as marketing, software and service development,

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

Figure 1 – Taxonomy of Conflicting Elements

product design, manufacturing and so on. The heterogeneity of the teams and disciplines in PSS development creates a higher number of conflicts.

Every team has a particular perspective on the system. Teams employ different tools, workflow processes and methodologies to realize their objectives. Even the language that a team uses is influenced by domain-specific terminologies. Such factors lead to conflicts during collaborations. To reach a comprehensive overview, we reviewed the state of research and practice. We formulated the findings into the taxonomy of conflicting elements during requirements management (see Figure 1).

The taxonomy is of high managerial importance as it increases the knowledge of human and non-human project factors that can create a conflict. Such a knowledge allows managers to do a better decision making in complex situations. By identifying the elements at the early stages of the PSS development, managers reach a deeper understanding on the extent to which the project is heterogeneous. The more various elements exist, more conflicts are likely to emerge and more complexity will the project face. Accordingly, the managers would be able to (1) prepare for dealing with the potential identified conflicts based on the taxonomy (2) increase everyone's awareness with regard the elements, which increase the complexity of the project. Accordingly, different strategies can be taken by the managers. For example, differences among teams and background of the teams can be exploited as a source of creativity and innovation or by limiting the collaboration among different teams, we follow an avoidance strategy.

IDENTIFYING DEPENDENCIES AMONG REQUIREMENTS SPECIFICATIONS

A PSS brings together functions of a product with aims of various services. Consequently, to design and develop a PSS, the PSS provider must consider requirements of a very heterogeneous set of hardware, software and service components. Numerous components of a PSS are interdependent and so are their requirements. For example, providing a new service implies new changes in software and hardware components, which should be handled by people from different teams and probably departments. In the design phase, we require interdependencies among the specifications in order to detect potential inconsistencies. During change management, knowledge of requirements interdependencies facilitates efficient change propagation. Similarly, in order to define test cases, which cover all aspects of the PSS, we need to know how different components are related based on their requirements.

Nevertheless, requirements specifications are mostly written in natural language or semi-formal format, which means they include natural language text to some extent. This is due to the fact that various stakeholders with different background knowledge use the requirements specifications as a means of communication.

Natural language is the only format that everyone can easily understand and relate.

However, the inherent complexity and ambiguity of natural language makes it very challenging for automatic analysis by machine. In the case of this research, the goal is to identify automatically a particular type of dependency among specifications or in another words natural language sentences.

To this end, we employed a machine learning approach as there it has a high potential due increasing capabilities of machine learning algorithms, availability of data and advances in the field of natural language processing. Moreover, a machine learning approach allows every company and project to adopt the solution to their own specific situation and needs.

With such a vision, we designed and implemented the solution. We prepared a training set of dependent requirements. Using natural language processing libraries, we extracted a set of features from every sentence and their relationships. We analyzed performance of different learning algorithms. The focus of our evaluation was on whether the developed solution is able to find specifications, which constrain each other or not. We argue that there are many different dependency types and every manager can define a new customized one. However, we chose this type of dependency because of its generalizability and relevance to any domain and settings.

The results show a great potential in this approach. We could reach a precision of about 90% and recall of about 75% for one learning algorithm. We suggest this solution for problems in which high precision is required, but missing several dependencies do not lead to any major faults. In contrary, we could reach about 90% recall and about 50% precision with another learning algorithm. Hence, in high sensitive contexts such a solution can be used. Moreover, we aim to collect user feedbacks on the identified dependencies. In this way, the solution would be able to improve its performance during its continuous usage.

As the next step, we are enhancing the solution with a graphical user interface and advanced graph analysis. For instance, one can easily find which requirement must be implemented first as a high number of other requirements are dependent on it. This facilitates the requirements prioritization process significantly. As the solution works with natural language specifications, it can be easily incorporated in every team and phase of PSS development.

Berkovich, M., Leimeister, J. M., Hoffmann, A., and Krcmar, H. 2014. "A Requirements Data Model for Product Service Systems," *Requirements Engineering* (19:2), pp. 161-186.

Nuseibeh, B., and Easterbrook, S. 2000. "Requirements Engineering: A Roadmap," *Proceedings of the Conference on the Future of Software Engineering*: ACM, pp. 35-46.

A4 Subproject Leaders:
Prof. Dr. Helmut Krcmar
krcmar@in.tum.de
Dr. Markus Böhm
markus.boehm@in.tum.de

Chair for Information Systems
Technical University of Munich
Boltzmannstr. 3
85748 Garching
Germany