Software-Supported Product Backlog Prioritization in Scrum Software Development Projects

Kleophas Model[1,*], Georg Herzwurm[1,**]

[1]University of Stuttgart, Information Systems II, Keplerstr. 17, 70174 Stuttgart, Germany

Abstract

Agile software development methods (e.g., Scrum) have gained increasing significance in recent years. In contrast to traditional plan-driven methods, they offer the necessary flexibility to react to continuously changing requirements. As the most established agile development method, Scrum involves the Product Owner (PO) as the sole representative of all project relevant stakeholders. A central task of the PO is the management of the Product Backlog (PB) that contains all relevant information to maximize the product value, i.e., requirements in the form of user stories. During PB management, creating a shared understanding of the underlying requirements is necessary. However, the PO is often a bottleneck, e.g., due to requirements complexity or limited access to relevant stakeholders. Especially the task of PB prioritization appears to be a challenging while success critical one. Appropriate software support can help to overcome challenges such as communication and collaboration barriers, complex calculations, or in-transparency regarding the prioritization process and result. While much research on manual or (semi-)automated prioritization techniques has been conducted, no software-supported PB prioritization currently exists that is based on a comprehensive methodological approach. The PhD project addresses this knowledge gap in developing a conceptual model and a therewith-aligned software prototype.

Keywords

scrum, agile software development, requirements prioritization, product backlog, software support

CEUR-WS.org/Vol-3316/phd-paper2.pdf

1. Problem & Motivation

In the context of software-intensive business, there is a tendency to ever shorter and faster development cycles due to an unpredictable environment with continuously changing requirements [1, 2]. In contrast to traditional plan-driven methods (e.g., the waterfall model or the V-model), agile methods (e.g., Scrum) promise to offer the necessary flexibility to react to changes in requirements [3]. An annual study by Komus et al. [4] reveals a remarkable decline in the use of plan-driven methods and, therefore, a respective increase in the use of agile methods.

Several agile methods have emerged based on the values and principles of the agile manifesto [5]. Since a complete specification of product requirements is often not possible at the beginning of the project or does not make sense due to the volatility of requirements, project execution

ICSOB'22: International Conference on Software Business, November 08–11, 2022, Bolzano, Italy

*PhD student and corresponding author.

**Supervisor of PhD student.

kleophas.model@bwi.uni-stuttgart.de (K. Model); georg.herzwurm@bwi.uni-stuttgart.de (G. Herzwurm)

https://www.bwi.uni-stuttgart.de/en/institute/team/Model/ (K. Model);

https://www.bwi.uni-stuttgart.de/en/institute/team/Herzwurm/ (G. Herzwurm)

0000-0002-9318-8369 (K. Model); 0000-0003-4663-0940 (G. Herzwurm)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

is started even without a complete specification [6]. Therefore, an iterative and incremental procedure is necessary. In each iteration, a subset of requirements is chosen to develop a potentially shippable increment that can be used for review and feedback collection. A continuous collaboration with relevant project stakeholders is pursued [7, 8]. It is obvious that not all product requirements can be implemented in a single iteration. Therefore, a comprehensible selection of appropriate requirements to be implemented is necessary. Thus, requirements prioritization is a central part of agile methods [9]. Without prioritization, there is the risk that unimportant requirements will be implemented in the next iteration, and more important requirements will be unconsciously postponed. Under certain circumstances, this can lead to project failures, which is why requirements prioritization is considered a critical success factor within agile software development projects [10].

The prioritization of requirements in the context of agile software development orients towards maximizing customer satisfaction, taking into account the first principle of the agile manifesto as it is "the highest priority to satisfy the customer" [5]. Nevertheless, the interests of all relevant stakeholders (including customers) need to be taken into account during prioritization [11]. As the most established agile software development method, Scrum involves the Product Owner (PO) as the sole stakeholder representative. It is his job to "represent the needs of many stakeholders" [11]. Scrum contains the product backlog (PB) to maintain, among others, stakeholder needs and product requirements, usually in the form of user stories [11]. These user stories represent short descriptions of requirements and how they are intended to fulfill the needs of specific stakeholders [12]. Moreover, the PB is an ordered list of user stories and other items (e.g., features or bugs). As the PO is responsible for PB management, it is his job to bring these items in the correct order according to their priority and thus to "maximize the value of the product resulting" [11]. To establish a shared understanding, the PO decisions must be respected by the stakeholders and represented in the PB. Therefore, one central task of the PO is the continuous PB prioritization that includes an ongoing consolidation, incorporation, and assessment of relevant stakeholder input [11].

However, the PO often becomes a 'bottleneck' and thus hinders PB management [13]. There are several reasons for this. First, the high number of divergent requirements often leads to complexity in the prioritization subject, while less-documented user stories make continuous communication necessary to dissolve complexity. Second, the high number of divergent stakeholders with divergent interests leads to complexity in decision-making. In addition, these stakeholders are not directly personally accessible (e.g., due to distributed projects or remote-work settings) [14]. Especially in scaled agile frameworks, such as *Large Scale Scrum (LeSS)* or *Scrum of Scrums*, this problem intensifies. Here, additional coordination mechanisms between several Scrum teams, with several POs and several PBs, become necessary [15]. Third, Scrum leaves it open how to carry out the prioritization tasks as this depends on specific project context and situation [8]. Many established prioritization techniques can also be applied in Scrum. However, this often results in a 'technique jungle' in which orientation without having situation-specific guidelines on integrating techniques into Scrum is complex [14]. Last, no unified definition of the 'value' and its underlying criteria according to which the prioritization tasks have to be carried out exists [8]. These problems require methodological support to dissolve the PO bottleneck. Furthermore, appropriate software support, on the one hand, can help to overcome communication and collaboration barriers and, on the other hand, enable

the development of innovative PB prioritization approaches (e.g., data-driven prioritization). Within the research disciple of Information Systems, this need follows the understanding of an information system as a socio-technical system, which requires a bilateral alignment with/of underlying processes [16, 17].

2. State of Research & Knowledge Gap

The design of a methodological approach requires a complete description of what (i.e., activities) needs to be done by whom (i.e., roles) in a given situation (i.e., situation and context factors) with specific (software) tool support and how to ensure quality regarding process and results [18]. This understanding of a comprehensive methodological approach serves as a frame of reference for analyzing the current state of research and identifying the knowledge gap.

Within the literature, there is a multitude of manual prioritization techniques (e.g., *AHP*, *MoSCoW*, or *Numerical Ranking*) most of which originate from classical requirements engineering. Many of these manual prioritization techniques are complex and thus time-consuming and dependent on expert knowledge. Furthermore, only a few techniques support continuous re-prioritization after requirements changes [19]. To resolve challenges like calculation complexity and collaboration barriers, there are (semi-)automated advancements of classical prioritization techniques by software support, for example, *RedCCahp*, an AHP advancement that automatically checks for consistency in redundant requirement comparisons, or *CBRanking*, a machine learning algorithm to predictively generate a requirements ranking [14]. However, these techniques are relatively generic in terms of their application in specific project environments. Significantly, there is no information regarding their applicability in agile development methods (i.e., Scrum). Moreover, these techniques mainly lack a comprehensive methodological framework.

Bakalova et al. [8] developed a conceptual model for agile requirements prioritization. This model is based on empirical insights from practice and consists of eleven components (e.g., Project context, Prioritization criteria, or Developers' input). Furthermore, they conducted a literature review to map existing prioritization techniques to the conceptual model. In doing so, they identified central gaps regarding the suitability of these techniques and the conceptual model. Especially, there is a shortcoming regarding customer-orientated prioritization. Based on their findings, they call for further empirical investigations using the conceptual model proposed. However, appropriate software support is not recognized in the conceptual model yet. Moreover, it does not consider Scrum-specific factors.

Bakalova et al. [8], for example, stated that *Quality Function Deployment (QFD)* is one of the few prioritization techniques that is based on the principle of customer-centered product development. QFD goes beyond a prioritization technique. In its origin, QFD is a method for customer-centered product development to cause high quality. Schockert [20] developed Agile Software QFD, an adaptation of QFD that can be applied in agile software development projects. Although it represents QFD for agile methods in general, it is mainly based on Scrum principles. Moreover, it is possible to integrate further techniques into Agile Software QFD to enable flexibility in situation-appropriate applications. Schockert [20] stated that it is necessary to conduct an empirical evaluation and to implement Agile Software QFD in a software tool.

Rietz & Schneider [13] present target-means statements (i.e., design principles) for a cooperative requirements prioritization system based on the theory of shared understanding. It is mainly based on the MoSCoW technique. However, it lacks empirical problem identification as the design principles are derived from issues elaborated in the context of a literature review.

Besides work on software-supported PB prioritization in science, there are several software tools in practice, either for PB management in general or specifically for PB prioritization. *Atlassian Jira* is one of the most established commercial software tools for PB management. *Ducalis.io* provides a broad toolbox of different prioritization techniques implemented in a web application. Besides the fact that these software tools are not scientifically founded, they do not represent a methodological approach.

Summarizing, to the best of the authors' knowledge, there currently exists no software-supported PB prioritization for Scrum software development projects that is scientifically based on a comprehensive methodological approach. This assumed knowledge gap is the starting point for the presented PhD project.

3. Research Design & Methodology

Pursuing an explorative, design-oriented approach, the PhD project aims to fill the previously derived knowledge gap. Therefore the research objective is the conceptualization of a software-supported PB prioritization for Scrum software development projects that is scientifically based on a comprehensive methodological approach. The central part of the research objective is developing a conceptual model that contains practically usable target-means statements for PB prioritization (esp. regarding methodological implementation and its support through a software tool). Conceptual models form a reconstructive abstraction of purposeful concepts for structuring and presentation of complex information systems. The models have both descriptive and prescriptive purposes and therefore pursue the goal of a novel solution [21, 22].

The research gap results in the need for software support to ensure the feasibility of the methodological approach. Therefore, in addition to the conceptualization of the methodological approach, the conceptualization of the software tool is a central part of the research objective. Following the purpose of classical business-IT-alignment, this research approach accompanies the bidirectional interrelationship of business (i.e., requirements that stem from organizational structures) and IT (i.e. technological solution characteristics). Therefore, on the one hand, it is necessary to align the software architecture with the process architecture. On the other hand, to promote innovative research results and exploit the existing technological potential of software, it is necessary to allow enablement regarding the methodological approach through possible software support [17, 16]. To address the bidirectional interrelationship of business requirements and technological solution characteristics, a continuous evaluation of both the underlying methodological concept as well as the technical solution are necessary. Thus the instantiation of the conceptual model as a software prototype is a mandatory result of the PhD project. This results in the following **Main Research Question:** *How can a software-supported product backlog prioritization for Scrum software development projects be designed?*

To reach the research objective, the PhD project follows the Design Science Research Methodology (DSRM) according to Peffers et al. [23] consisting of six process steps. The DSRM allows

research projects to develop innovative IT artifacts based on existing science problems and real-world practice problems. In addressing the problem and solution space, the DSRM is recommended in the context of explorative, design-oriented research projects [24].

The core component of the conceptual model to be developed is a methodological approach integrated into Scrum. Therefore the DSR project follows the situational method engineering (SME) methodology that allows the construction and/or configuration of a method (also including techniques and tools) by applying engineering activities. As the configuration of a method depends on specific situation and context factors, SME requests the construction of situation-specific method configurations [18, 25]. SME has been established as a proven research method for tailoring agile methods [26]. The main idea of SME is developing an extensive method base that allows the selection of appropriate method fragments that fit the situation. Having assembled and applied the method, its performance needs to be measured and evaluated so that an iterative reconstruction of the method base and the method fragment selection can be proceeded [18].

Since the research objective focuses on developing practically usable target-means statements, a problem-centered initiation of the DSR project that addresses challenges in practice is necessary. Therefore, by applying DSRM step 1 *(Problem identification)*, the research problem is further motivated from a practical perspective. The resulting challenges serve to define objectives of the software-supported PB prioritization to be further developed [23] in context of **Research Question 1:** *What challenges exist in practice with regard to PB prioritization in Scrum?*

To answer this research question, an empirical qualitative cross-sectional analysis of methodological approaches with regard to PB prioritization in Scrum software development projects is conducted. For this purpose, the experience of practice experts (especially but not exclusively POs, Scrum Masters, Software Engineers, Requirement Engineers, Business Analysts, Product Managers, and customers) will be analysed in semi-structured, guideline-based expert interviews [27]. For a thorough elicitation of problems and the associated derivation of requirements (see research question 2), the interviews are supplemented by the Critical Incidents Technique [28, 29]. Based on the identified challenges, in context of DSRM step 1, in DSRM step 2 *(Objective definition)* it is necessary to transform these descriptive challenges into normative design objectives. These design objectives, in the sense of solution-neutral requirements, need to be fulfilled by the targeted solution. Therefore, these requirements are the guiding basis for the conceptualization and the subsequent evaluation of the software-supported PB prioritization [23]. This objective results in **Research Question 2:** *What are the requirements on a software-supported PB prioritization in Scrum software development projects?*

On the one hand, the requirements can be directly obtained in the context of the cross-sectional analysis (i.e., expert interviews). On the other hand, an argumentative-deductive analysis of the challenges is carried out, resulting in the derivation of corresponding requirements. The derived requirements have a hypothetical character and must therefore be subjected to evaluation.

DSRM step 3 *(Design & Development)* provides for the design and development of justified solution artifacts, e.g., in the form of methods and/or models, that address the requirements previously derived. Therefore, the goal is to propose well-argued solution characteristics and bring them into a comprehensible relationship. The development of the solution artifact thus anticipates, on the one hand, the elicitation of appropriate solution characteristics and, on the other hand, the synthesis of these separate solution characteristics into a systemic solution

construct, i.e., a conceptual model. Therefore, prescriptive target-means statements that bring solution characteristics and requirements into a comprehensible relationship form the core of the conceptual model [23]. This results in **Research Question 3:** *Which solution characteristics should a software-supported PB prioritization for Scrum software development projects contain?*

SME envisages establishing a method based on already existing and/or newly developed methods [18]. Identifying existing potential solution features takes place in a qualitative secondary data analysis. Triangulation of sources and methods is conducted for collecting these method components, which are relatively diverse in terms of their types. For the identification of methodological approaches and techniques, an analysis of scientific and grey literature will take place. A document and tool analysis is performed to identify and analyze the functionality of existing software tools. Following the design-oriented research design that aims for innovative artifacts, well-justified new solution characteristics should be proposed and integrated into the methodological basis. This will be done in the context of an argumentative-deductive analysis taking into account the identified requirements. The SME step of assembling the method fragments is performed by argumentative-deductive analysis. At this point, the target-means relationships are formed, and thus the design recommendations are derived. The connections between solution characteristics and requirements are represented within a QFD prioritization matrix from which target-means statements can be derived. Finally, a conceptual model is developed that integrates the single solution characteristics into a systemic relationship [22, 21].

To make the conceptual model tangible and thus accessible for evaluation purposes, this DSRM 4 step *(Demonstration)* contains an instantiation of the conceptual model as a functional software prototype in the sense of a minimum viable product (MVP). This instantiation is understood as a proof-of-concept and is thus already a central part of the evaluation [30, 23, 31]. Besides the demonstration of functional feasibility, proof-of-concept evaluation intends to deepen the understanding of the addressed problem space, allow the generation of scholarly knowledge to further refine the solution characteristics or addressed requirements [30].

Besides proof-of-concept evaluation, Nunamaker et al. [30] propose proof-of-value evaluation as a second stage. It primarily aims to demonstrate the efficacy of an artifact, meaning to investigate and measure causal relations between solution characteristics and the underlying requirements. Moreover, proof-of-value evaluation is intended to comprehend influencing factors that affect the application of the proposed solution, e.g., situational context factors. Therefore, proof-of-value evaluation corresponds with the DSRM step 5 *(Evaluation)* [23].

The evaluation of the identified challenges (research question 1) and the requirements (research question 2) is to be carried out within the context of an empirical qualitative cross-sectional analysis using an online survey. On the one hand, practice experts are invited to assess the relevance of the challenges based on their experience. On the other hand, the practice experts have to evaluate the identified requirements in terms of their importance using the 100 dollar method, a specific implementation of cumulative voting, so that prioritization of the requirements can occur afterward [32].

The evaluation of the conceptual model, in particular the underlying solution characteristics and design recommendations (research question 3), takes place in two stages. In the first step, practice experts are invited to use the prototype to perform a constructed prioritization scenario. In doing so, the study participants are asked to articulate their thoughts aloud while using the prototype ('thinking aloud') [33]. In the second step, the participant is asked to fill out a

questionnaire that covers a satisfaction survey combining Likert scaled and free text questions. Based on this, a Kano classification of the solution features can take place afterward [34].

According to DSRM step 6 *(Communication)* ongoing communication of the research results at relevant scientific conferences (e.g., ICSOB) and in practice communities (e.g., ISPMA) are planned [23].

4. Timeline & Preliminary Results

The overall duration of the PhD project is planned as four years. In the first year, the research problem and the research design were elaborated. Moreover, two empirical pre-studies have been conducted to verify the (assumed) practical research gap and to evaluate the questionnaire for the expert interviews. These pre-studies identified an initial set of challenges, requirements, and solution characteristics. The findings confirmed the practical research gap and, thus, the relevance of the PhD project. In the second year, a case study was initiated with a big and prominent German software company (case A). Ten interviews were conducted with product managers, POs, developers, and team leads to identify further challenges and requirements. In parallel, the first version of a web application that is based on SpringBoot and Angular was implemented based on the identified requirements to that date. In another case study with the software department of a big German insurance company (case B), this prototype could be evaluated, and further challenges, requirements, and solutions requirements could have been elaborated. The main idea of the solution concept is based on Agile Software QFD by Schockert [20]. Thus, the separation of stakeholder needs and solution characteristics is pursued. This is achieved by establishing dedicated backlogs for stakeholders, stakeholder needs, and product functions. The content of these backlogs flows in the final PB. The prioritization of PB items is done by cooperatively prioritizing the different backlog items according to stakeholder competencies and responsibilities. These preliminary results were presented at IWSiB'22 [35].

For the next remaining two years the following activities are planned: The prototype will be further developed. Soon, there will be thinking-aloud tests and satisfaction surveys with experts of case company A conducted. Moreover, a global (i.e., separated from specific case studies to gain a high number of returns) online survey is conducted to assess the relevance of challenges and to prioritize requirements.

5. Expected Contributions

According to design-oriented information systems research, the expected research contributions can be classified into descriptive, normative, and prescriptive results [23]. These results are intended to contribute to the identified knowledge gap and to research mainly in the context of software-intensive business, i.e., in the areas of agile software development, and requirements engineering [1]. As the primary research objective is developing scientifically founded but also practically usable target-means statements, besides science, the PhD project contributes to practice in solving real-world practice problems. Therefore the research results also address companies in the context of software-intensive business, e.g., software development projects that apply the Scrum process model and need support in prioritizing the PB.

By answering research question 1, challenges that exist in real-world practice are identified. This allows for creating a problem-oriented knowledge base for the PhD project and further research endeavors. To solve these identified challenges, well-argued solution-neutral requirements are collected in a requirements backlog that serves as the normative objective for further design science research projects especially starting with an objective-centered initiation. Finally, solution characteristics related to the derived requirements in the form of prescriptive target-means statements serve as design knowledge for future instantiations of the conceptual model to pursue empirical studies in science and improve real-world Scrum projects in practice.

References

[1] P. Abrahamsson, J. Bosch, S. Brinkkemper, A. Mädche, Software business, platforms, and ecosystems: Fundamentals of software research, 2018. doi:10.4230/DagRep.8.4.164.

[2] J. Highsmith, A. Cockburn, Agile software development: The business of innovation, Computer 34 (2001) 120–122. doi:10.1109/2.947100.

[3] K. Petersen, C. Wohlin, The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study, Empirical Software Engineering 15 (2010) 654–693. doi:10.1007/S10664-010-9136-6/FIGURES/5.

[4] A. Komus, M. Kuberg, Ergebnisbericht: Status quo (scaled) agile 2019/20, 2019.

[5] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, et al., Manifesto for agile software development, 2001.

[6] D. Port, T. Bui, Simulating mixed agile and plan-based requirements prioritization strategies: proof-of-concept and practical implications, European Journal of Information Systems 18 (2009) 317–331. doi:10.1057/ejis.2009.19.

[7] ISO/IEC/IEEE 2621525:3023, Systems and software engineering - Developing user documentation in an agile environment, Technical Report, ISO, Geneva, CH, 2012.

[8] Z. Bakalova, M. Daneva, A. Herrmann, R. Wieringa, Agile requirements prioritization: What happens in practice and what is described in literature, in: Lecture Notes in Computer Science 6606, 2011, pp. 181–195.

[9] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Information Systems Journal 20 (2010) 449–480.

[10] K. Curcio, T. Navarro, A. Malucelli, S. Reinehr, Requirements engineering: A systematic mapping study in agile software development, Journal of Systems and Software 139 (2018) 32–50. doi:10.1016/J.JSS.2018.01.036.

[11] K. Schwaber, J. Sutherland, The scrum guide, 2020.

[12] M. Cohn, User stories applied: For agile software development, Addison-Wesley, 2004.

[13] T. Rietz, F. Schneider, We see we disagree: Insights from designing a cooperative requirements prioritization system, in: Proceedings of the 28th European Conference on Information Systems (ECIS 2020), 2020.

[14] F. Hujainah, R. B. A. Bakar, A. B. Nasser, B. Al-haimi, K. Z. Zamli, Srptackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects, Information and Software Technology 131 (2021) 106501.

[15] D. Šmite, N. B. Moe, A. Šāblis, C. Wohlin, Software teams and their knowledge networks in large-scale software development, Information and Software Technology 86 (2017) 71–86.
[16] J. M. Leimeister, Einführung in die Wirtschaftsinformatik, volume 13, Springer, 2021.
[17] H. Krcmar, Informationsmanagement, volume 6, Springer Berlin Heidelberg, 2015.
[18] S. Brinkkemper, Method engineering: engineering of information methods and tools, Information and Software Technology 38 (1996) 275–280.
[19] P. Achimugu, O. Achimugu, M. A. Taiye, S. Husssein, G. Tam-Nurseman, S. Adekeye, How to support communication among stakeholders during software requirements prioritization, Journal of Software Engineering and Applications 14 (2021) 267–276.
[20] S. Schockert, Agiles Software Quality Function Deployment, Eul Verlag, 2017.
[21] U. Frank, S. Strecker, P. Fettke, J. Vom Brocke, J. Becker, E. Sinz, The research field "modeling business information systems", Business & Information Systems Engineering 6 (2014) 39–43.
[22] Y. Wand, R. Weber, Research commentary: Information systems and conceptual modeling—a research agenda, Information Systems Research 13 (2002) 363–376.
[23] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of management information systems 24 (2007) 45–77.
[24] A. R. Hevner, S. T. March, J. Park, S. Ram, Design science in information systems research, MIS Quarterly 28 (2004) 75–105.
[25] B. Henderson-Sellers, J. Ralyté, P. J. Ågerfalk, M. Rossi, Situational method engineering, Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-642-41467-1.
[26] A. S. Campanelli, F. S. Parreiras, Agile methods tailoring - a systematic literature review, Journal of Systems and Software 110 (2015) 85–100. doi:10.1016/j.jss.2015.08.035.
[27] A. Bogner, B. Littig, W. Menz, Interviewing Experts, Palgrave Macmillan Ltd., 2009.
[28] J. L. Gogan, M.-D. McLaughlin, D. Thomas, Critical incident technique in the basket., in: Thirty Fifth International Conference on Information Systems (ICIS 2014), 2014.
[29] J. C. Flanagan, The critical incident technique, Psychological Bulletin 51 (1954) 327–358.
[30] J. F. Nunamaker, R. O. Briggs, D. C. Derrick, G. Schwabe, The last research mile: Achieving both rigor and relevance in information systems research, Journal of Management Information Systems 32 (2015) 10–47. doi:10.1080/07421222.2015.1094961.
[31] E. Ries, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Sucessful Businesses, Crown Business, 2011.
[32] D. Leffingwell, D. Widrig, Managing software requirements: a unified approach, Addison-Wesley Professional, 2000.
[33] K. A. Ericsson, H. A. Simon, How to study thinking in everyday life: Contrasting think-aloud protocols with descriptions and explanations of thinking, Culture and Activity 5 (1998) 178–186.
[34] K. Matzler, E. Sauerwein, The factor structure of customer satisfaction, International Journal of Service Industry Management 13 (2002) 314–332. doi:10.1108/09564230210445078.
[35] K. Model, C. Mombrey, G. Herzwurm, Paving the way to a software-supported requirements prioritization in distributed scrum projects, in: 2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB), 2022, pp. 51–58.