PDISC – Towards a Method for Software Product DISCovery

Type: Exploratory Paper

Karl Werder[1(✉)], Benedikt Zobel[2], and Alexander Maedche[3]

[1] University of Mannheim, Mannheim, Germany
werder@es.uni-mannheim.de
[2] Osnabrueck University, Osnabrueck, Germany
benedikt.zobel@uni-osnabrueck.de
[3] Karlsruhe Institute of Technology, Karlsruhe, Germany
alexander.maedche@kit.edu

Abstract. For the creation of software products, the idea of iterative and incremental development and design is widely accepted and embedded in various methodologies. However, earlier activities within software projects are often the cause for the projects termination. Such activities are often described as the product discovery phase. Therefore, this study develops PDISC, a method for software product discovery. Following a design science research approach, a systematic literature review extracts design requirements and method fragments from literature. The method fragments describe early activities and are documented using process deliverable diagrams. Collectively, such method fragments form a method database that is used to develop PDISC. PDISC helps practitioners to conduct early activities in a systematic way in order to create a product vision.

Keywords: Software product · Discovery · Method engineering · Product vision

1 Introduction

Product discovery phase is the term used to describe early activities collectively in order to create a viable, desirable and feasible product vision. These early activities provide a different set of challenges and our understanding of their precise influence on a product remains unclear [1, 2]. Hence, practitioners require actionable guidance in the form of a method for the discovery of software products [1, 2]. Such method can help practitioners to structure their early activities. Moreover, it assures the correct shaping and documentation of the product idea in the form of a product vision.

More recently, scholars explore the combination of different methodologies. For example, the combination of agile software development (ASD) and user-centered design (UCD), i.e. blending practices and techniques for development with those established in the design discipline (e.g. [1, 3, 4]). Fox et al. [3] for example, suggest a method combining ASD and UCD through a cycle zero and parallel yet interwoven tracks. Focusing on the individual, da Silva [5] worked extensively to identify the role of

A. Maglyas and A.-L. Lamprecht (Eds.): ICSOB 2016, LNBIP 240, pp. 47–62, 2016.
DOI: 10.1007/978-3-319-40515-5_4

UX designers within agile teams and the integration of interaction design into ASD [6]. In a similar vein, Ferreira, Noble, and Biddle [7] suggest steps towards the cooperation of user experience designers and agile developers. Brhel et al. [1] identified five principles along the processes and practices of UCD and ASD domains, establishing a user-centered agile software development approach. However, we lack systematic knowledge on how to conduct early activities and deliverables during the product discovery phase [1, 2]. The objective of such phase is the creation of a product vision that improves the software's success. To the author's knowledge, there is no discovery method for software products suggested in the literature.

Therefore, we follow calls for more research on product discovery [1, 2, 8] and seek to develop a method for software product discovery. Our research objectives are to: (a) review existing literature on methods regarding the discovery of software products, (b) extract and formalize existing methods from such literature to establish clear design requirements and design principles, and (c) translate those design requirements into a formalized method for software product discovery. We formulate the following research question: *How to design a method for a software product discovery phase?*

The paper contributes to practice and theory. The practical contribution is PDISC, a method for software product discovery. Such method helps practitioners to articulate needed activities and deliverables when discovering software products. In addition, it provides a checklist in the form of a comprehensive list of activities and deliverables during such process. While processes within firms may vary depending on situational factors, such list creates awareness of fundamental activities and deliverables of product discovery. The theoretical contribution of the paper is the systematic extraction of design requirements and subsequent formulation of clear design principles. These principles categorize the requirements along product-, user- and team-related aspects and therefore, address concerns of viability, desirability and feasibility of the envisioned product. We establish clear phases within product discovery that form a framework for future research.

2 Foundations and Related Work

The term product discovery has been heavily used in the pharmaceutical domain and the area of drug discovery (e.g. [9]). However, in recent years the term is used to describe a phase of upfront activities preceding the product development and product design phases [1]. In the field of new product development, it describes the ideation generation stage [10]. Others highlight the importance of this phase to determine the actual need for such a product and the existence of a user base on the one hand, and the actual feasibility of such a solution on the other hand [11].

As shown in Fig. 1, ASD is one possible representation of product development phase, while UCD serves as representation of product design. Product design is a key element in various software development methodologies. Some authors describe it as conceptualization of a solution prior to programming activities [12]. Product development, in turn, describes the creation or implementation of software artifacts, e.g. by programming [13]. In iterative or agile frameworks these two phases are executed multiple times, and are thus depicted as parallel to one another (e.g. [1, 3]).

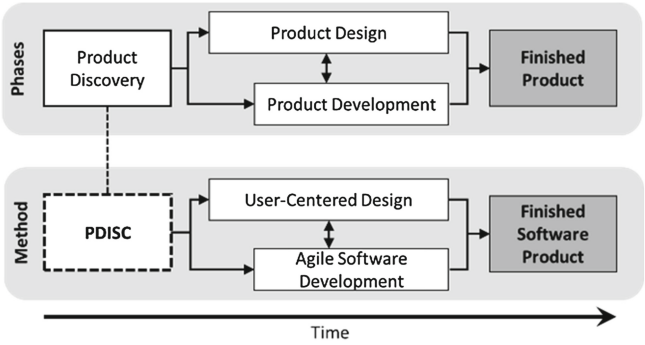

Fig. 1. Placement of a method for product discovery in existing methods and phases

The combination of ASD and UCD becomes more and more a research stream by itself. In ASD and UCD, the strict separation of upfront and development and design phase (as it is done e.g. in waterfall-models) was changed to an iterative process, leaving early activities mostly out of scope [1, 3, 12, 14]. Scrum as methodology proposing concrete guidelines for agile development in formulating teams and roles also does not focus on activities prior to building software artifacts [15, 16]. More insights could stem from the not software-related areas of New Product Development and Innovation Management. In these, Product Discovery was introduced as so called front-end phases, with buzzwords such as "fuzzy front-end" and "front-end innovation" [17–19]. The key goal of these phases is to reduce uncertainty and equivocality that is largely present at the early stages of product development [17–21]. Therefore, a method summarizing and structuring early activities related to software development and design is entitled PDISC.

3 Research Method

For the development of the software product discovery method, we opt for a design science research (DSR) approach [22, 23] in combination with the discipline of method engineering [24–26]. The study investigates the method for product discovery as its central artifact. Moreover, the study aims at solving a practical problem by designing an appropriate artifact [27]. First, the problem is identified (cf. Sect. 1). In order to define the solutions objective, the study reviews the literature, defines the term software product discovery and extracts design requirements. Next, the study develops design principles that guide the design and development of PDISC. For the design and development, the study relies on a method database.

In order to extract design requirements and method fragments, the study starts with a systematic literature review (SLR) [28, 29]. A systematic process and transparent documentation of the literature allows the reader to assess the completeness of the review [29]. Hence, the first step in conducting a SLR is the development of a study protocol. The protocol documents the main research questions, key decisions along the scope (e.g. search strategy; databases; inclusion, exclusion, and quality criteria), and a concept-matrix. Figure 2 presents the search strategy. Hereafter, the studies inclusion

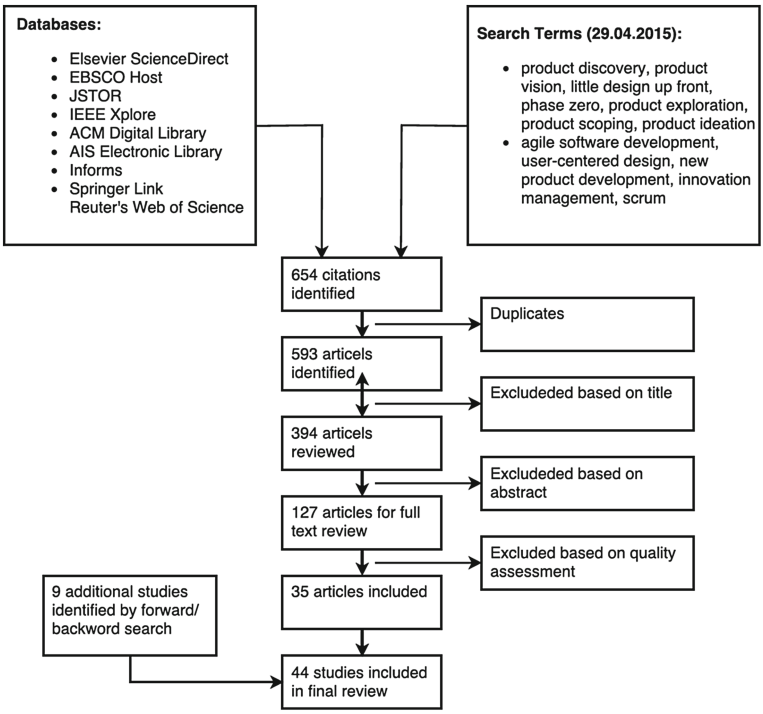

Fig. 2. Search strategy diagram

and exclusion criteria, the data sources and search strategy, and the data extraction and analysis process are described in more details.

3.1 Inclusion and Exclusion Criteria

The study includes articles that provide insights on potential shortcomings w.r.t. the product discovery phase of software products. In addition, articles that describe possible solutions and/or recommendations of a product discovery phase are included. The investigated timeframe is from 1997 until mid-2015, given the origins of agile as an established reference methodology. After the initial query of the databases, the exclusion of duplicates reduces the initial set of 654 articles to 593 articles. Thereafter, the exclusion based on the title, e.g. with an irrelevant industrial focus such as biology and pharmacy or industrial engineering, leads to a selection of 394 articles. Following, reading the abstracts excludes those articles that do not focus or contribute to product discovery. Consequently, 127 articles remain for further assessment. Based on this assessment, every paper receives a score on a 5-point Likert scale assessing their applicability to four questions. The questions assess whether a goal is mentioned by the authors, whether there is an empirical part of this paper, whether shortcomings or challenges related to product discovery are mentioned and whether recommendation to product discovery are documented. The study excludes articles that receive a score of less than three. As a result, 35

articles are relevant for this study. Conducting a forward and backward search leads to the inclusion of another nine articles, resulting in 44 articles.

3.2 Data Sources and Search Strategy

The sources focus on databases that include publications from the field of information systems discipline, computer science and general management. Hence, following a rather inclusive approach in response to some critics and therefore, also identifying less impactful yet still relevant publications in the search results [30]. We include the following nine databases: Elsevier ScienceDirect, EBSCO Host, JSTOR, IEEE Xplore, ACM Digital Library, AIS Electronic Library, Informs, Springer Link and Reuter's Web of Science. Given that different synonyms exist for the term product discovery, a preceding exploratory literature search identifies the keyword-matrix. In order to assure that the correct keywords are in the search string and the study identifies relevant articles, we conduct a pilot test using IEEE as the sample database. We analyze the resulting 134 articles for plausibility and soundness. Thereafter, the keyword-matrix builds the basis for the search string in order to identify and evaluate prevalent matches in research literature. Consequently, we formulated the following search string:

```
("product discovery" OR "product vision" OR "little design up
front" OR "phase zero" OR "product exploration" OR "product
scoping" OR "product ideation")
AND
("agile software development" OR "user-centered design" OR "new
product development" OR "innovation management" OR "scrum")
```

3.3 Data Extraction and Data Analysis

In order to extract information, the study analyzes 44 articles for their shortcomings and proposed actions related to product discovery. Clusters start to form and the articles build groups along identified commonalities (see Table 1). During the full text review, the authors highlight critical sections and aggregate them in order to transfer them into the research database. The database documents key information, such as the dimension, the corresponding activity, phase and methodology, and their use in design requirements and design principles. While 28 articles suggest shortcomings of current practices in a discovery phase, 30 articles provide suggestions on the implementation of early activities. Articles proposing activities are potential contributions towards the method database. Thereafter, nine articles with concrete and multiple activities form the method database. A Process-Deliverable-Diagram (PDD) models and documents the activities of each article. Documenting the activities using a PDD enriches the methods' understanding and structures the method fragments.

4 Results

From the final list of articles, we identify eight common challenges (see Table 1). Overall, we see three related groups of challenges, i.e. product-related, user-related and team-related. First, challenges related to the product, i.e. the need to define the product

Table 1. Challenges related to the discovery of products mentioned in primary articles.

Challenges	References
Product vision needs to be defined earlier and clearer that it is current practice	Ferreira, Noble, & Biddle, 2007 [7]; Heikkilä et al., 2015 [15]; Hildenbrand & Meyer, 2012 [30]; Hollis & Maiden, 2013 [31]; Kajko-Mattsson & Nyfjord, 2009 [32]; Kakar & Carver, 2012 [20]; Nyfjord & Kajko-Mattsson, 2008 [33]; Qumer & Henderson-Sellers, 2007 [34]; Sarpong & Maclean, 2012 [35]; Sibghatullah & Hussain, 2006 [36]; Stevens, 2014 [17]; Tessarolo, 2007 [37]; Vanhanen, Itkonen, & Sulonen, 2003 [38]
User involvement as early as possible	Brhel, Meth, Maedche, & Werder, 2015 [3]; Cloyd, 2001 [39]; T. S. Da Silva, Martin, Maurer, & Silveira, 2011 [8]; Ferreira et al., 2007 [7]; Fox, Sillito, & Maurer, 2008 [1]; Hildenbrand & Meyer, 2012 [30]; Kuusinen, 2014 [16]; Patton, 2002 [40]; Rejeb, Boly, & Morel-Guimaraes, 2008 [41]; Salah, Paige, & Cairns, 2014 [42]; Sibghatullah & Hussain, 2006 [36]; Sohaib & Khan, 2010 [14]
Conduct little design upfront	Adikari, McDonald, & Campbell, 2009 [43]; Brhel et al., 2015 [3]; T. S. Da Silva et al., 2011 [8]; Ferreira et al., 2007 [7]; Kuusinen, 2014 [16]; Miller, 2005 [44]; Salah et al., 2014 [45]; Salah, Paige, & Cairns, 2014 [42]
Utilize fuzzy front end and front end innovation	Frishammar, Florén, & Wincent, 2011 [18]; Kakar & Carver, 2012 [20]; Khurana & Rosenthal, 1998 [21]; Knoll & Horton, 2011 [46]; Oliveira & Rozenfeld, 2010 [47]; Rejeb et al., 2008 [41]; Sperry & Jetter, 2009 [19]; Stevens, 2014 [17]
Establish a sprint zero	Adikari et al., 2009 [43]; Heikkilä et al., 2015 [15]; Inayat, Salim, Marczak, Daneva, & Shamshirband, 2015 [48]; Kajko-Mattsson & Nyfjord, 2009 [32]; Loniewski, Armesto, & Insfran, 2011 [49]; Sibghatullah & Hussain, 2006 [36]
Developing a low-fidelity prototype	Cloyd, 2001 [39]; T. S. da Silva et al., 2011 [6]; Fox et al., 2008 [1]; Salvador, Nakasone, & Pow-Sang, 2014 [50]
Moving items from product vision to product backlog	Cloyd, 2001 [39]; Hildenbrand & Meyer, 2012 [30]; Qumer & Henderson-Sellers, 2007 [34]; Vanhanen et al., 2003 [38]
Using the concept of design thinking	Hildenbrand & Meyer, 2012 [30]

vision early and communicate its practice clearly. Second, when working with users, developers and designer face a common challenge, i.e. early user involvement. Organizations often delay such involvement, resulting in negative consequences. Third, teams need to adopt the correct practices, such as little design upfront and the development of low-fidelity prototypes.

5 Designing a Method for Software Product Discovery

Product Related Requirements. While ASD is an established method with a focus on the software's functionality, it is vague about the starting conditions. Examples are the desirable upfront-activities that are not planned in development projects [43]. As a result, these steps often lack time and budget during their execution [43, 46, 48]. Other appeals include calls for a clearer position of exploratory activities in software development projects, insufficiently addressed by current agile methods [52], as well as calls for less uncertainty and ignorance within pre-implementation phases of agile projects [34]. Furthermore, reflecting mentions criticize the implementation of operational planning activities only in later development phases, and a lack of describing activities required for the creation of product visions or product backlogs, even in cases the notion of these documents is used by scholars [15, 31]. Hence, we formulate the first Design Requirement (**DR1**): *A method for Software Product Discovery should articulate early activities that improve the software product development environment.*

Idea generation is a key activity in a proposed pre-phase 0 [21], executed prior to the development of a new product, and prior to the identification of detailed customer needs, technological capabilities, or core product requirements [17]. Menor, Tatikonda, and Sampson [53] describe a similar idea generating activity in the area of new service development. An initial ideation step as well as idea generation technique is also used in new product development [19, 47]. Therefore, we formulate (**DR2**): *A method for Software Product Discovery should collect initial product ideas.*

Some sources mention that agile practices only start after a vision or more concrete artifacts are established [31]. According to Sarpong and Maclean [36], a product vision can be defined as the "mental image of a yet to be realized product". Others try to conceptualize ways to reach a useful vision. An extended envisioning process is suggested as a way to identify high-level requirements [32]. More explicitly, the product vision serves as a key input element for all further activities [39]. Kajko-Mattsson and Nyfjord [33] even describe a product vision planning phase in detail, aiming at identifying a so-called product vision plan. Sibghatullah and Hussain [37] introduce the so-called product vision statement as a result of visioning activities, and state that the product vision becomes more important, the higher the uncertainty or complexity of the product goal is. Using the term "High-Level Product Scope", others propose the up-front activity of building a visionary scope initially, and revising it through all iterations to come [49]. In design thinking, the vision is frequently enriched in the ideate-phase [31]. Furthermore, counting towards a vision is the "holistic design vision" as a result of an iteration 0 [43]. As a result of a pre-phase 0 different kinds of a version can be possible,

for example for the business, a project, and a product simultaneously [21]. Kakar and Carver [20] state that the creation of a clearly defined product concept is important in order to effectively manage the software development process. Tessarolo [38] states that a product vision can further be important for on-time performance. The process of product vision planning is described by Nyfjord and Kajko-Mattsson [34]. A developed vision can be used for sharing a unified picture, for example, with the developers [54]. This in turn constitutes towards a unified understanding of the product in the project team [55]. Ferreira et al. [7] recommend that usability concerns should be a part of the vision, while Ebert [56] sees translated market needs as input for the vision. In order to include this upfront visioning work in the product discovery phase, we derive (**DR3**): *A method for Software Product Discovery should form a product vision.*

When taking a closer look at the results of the process, scholars mention that agile development practices should pay more attention to usability concerns. Literature often states that either pragmatic and hedonic qualities do not play a role at all [1, 3, 14, 41, 44], or only at a time much too late to have a major impact on the resulting product [40, 51]. One goal should thus be to include a user-focus during the process. This is supported by the statement that User-Centered Design would be "a perfect fit for an agile environment" [57], and that "requirements should be based on what users would be doing with the product". Hence, we derive (**DR4**): *A method for Software Product Discovery should extract users' needs for pragmatic and hedonic qualities.*

To improve software product development process and environment early activities have to be embedded and clearly articulated (DR1). A focus on the software product is broken down into elements, stating that a first instance collects initial product ideas (DR2). Such ideas will be developed into and form a concrete product vision (DR3). Furthermore, the two basic requirements of a software product, i.e. the software needs to be usable and useful are to be considered. As these four Design Requirements all share the focus of the software product itself, the first Design Principle (**DP1**) summarizes them: *Product context, goals, purposes and key requirements require clarification in order to improve product success*.

User Related Requirements. In ASD as well as UCD, users and customers play a larger role through a principle called "frequent stakeholder involvement" in comparison to traditional development or design techniques. Stakeholders are important in every development project. While there are many stakeholders available, the user is one of the most important ones. However, the creation process of a software product often neglects the user. For example, ASD does not provide concrete guidance on how to develop software that is user-friendly [43]. However, this poses difficulties during the development, especially in terms of software's usability. Often, a reason for the lack of usability is the low prioritization of usability during the development. In addition, there is often no guidance on user experience activities [16]. Some point out that the lack of user-focus in agile practices is a key reason for a lack in innovation management [42]. In order to cope with this issue, the literature presents different suggestions. One example is a condensed up-front user analysis [40, 45]. Therefore, we derive (**DR5**): *A method for Software Product Discovery should improve the adoption of practices for researching users' needs.*

Eventually, user requirements have to be collected and evaluated [58]. Hildenbrand and Meyer [31] mention that even mature processes, such as lean thinking do not provide descriptions on how to gain knowledge of user requirements. Sohaib and Khan [14] ask how user requirements could be gained from the stakeholders usually involved in agile feedback rounds. In most cases, this is the customer rather than the end-user. Sy [54] propose their version of a cycle zero, also incorporating user research. In order to process such requirements effectively, they need to a proper documentation. We identify (**DR6**): *A method for Software Product Discovery should properly document and record user requirements as a basis for further design and development activities.*

Building on the two prior design requirements, both, practices and documentation need to be integrated into the ongoing developments [50]. Even without starting actual programming activities, it is still necessary to prepare for a later phase. It ensures the ability to integrate collected requirements. For cycle zero, Sy [54] proposes the detailed inquiry of collecting data in order to support later phases. For example, the provision of exact target user descriptions. The process following the design thinking principles can also help to include the users' wishes into the finished product [31]. Other agile techniques, such as user-goal-analysis followed by prototyping activities, also try to improve this process [35, 40]. Following, we identify (**DR7**): *A method for Software Product Discovery should enable the integration of user requirements and user-centered practices into further design and development activities.*

The need to improve the adoption of user research practices (DR5) and the proper documentations of their results (DR6) provides a sound basis. However, these also need a successful integration into further design and development activities (DR7). As these requirements are user related and aim to include the user into the design and development activities, they are summarized into **DP2**: *A method of Software Product Discovery should provide methods to research and integrate specific users' needs and demands into the software product design and development process in order to the product's usability and user experience*.

Team Related Requirements. Furthermore, we find shortcomings of the overall understanding or "picture" by the design and development team. For example, da Silva et al. [8] points out that the "big picture" of what is expected is gone missing at some point in time throughout the project. This issue can be attributed to the fact that either there is not enough detail of the concepts to begin with, or the formulated product vision poses inconsistencies [33, 38, 39]. Countermeasures propose the constitution of a shared vision amongst all project members [18, 43, 46, 54, 59]. Hence, we suggest (**DR8**): *A method for Software Product Discovery should enable the product team to develop a unified understanding of the product.*

Hollis & Maiden [32] and Adikari et al. [44] further focus on requirements and requirements engineering, and how these processes lack creative thinking and a dedicated focus when implemented today. In current development projects, a lack of guidance can be observed that leads to either bad quality or longer development times [37]. The challenging combination of creative thinking and structured stepwise progress comes from fundamental difference between design and development. While the design emphasizes the creative part that can be hard to articulate and document, development

stemming from the engineering disciplines builds on stepwise and sequential process improvements. A design and development team needs to master both. Hence, we derive (**DR9**): *A method for Software Product Discovery should enable the product team to master both, development maturity and creative thinking.*

The lack of up-front activities as mentioned in the literature (e.g. [43, 46]) can be rooted back to a lack of management support and appreciation [52]. Consequently, a lack of other elements, such as maintenance, is found in the literature [33]. In describing general problems in ASD, Hollis & Maiden [32] state that the principle of simplicity found throughout agile processes has been taken too far to still provide any contribution towards innovation, while Ebert [56] finds fault with agile cycle times being too long to still be productive. In addition, the coordination in agile teams between the different functions leads to project delays [56]. Both, cycle time and project coordination are common management decisions. From a different perspective, Gamble & Hale [60] describe that scalability is difficult or problematic to achieve in ASD projects. Also, UCD is prone to management challenges. For example, Salah et al. [43] often find a lack of support by management roles towards user-centered activities, making it difficult for the team to execute them. Summarizing, we formulate (**DR10**): *A method for Software Product Discovery should assure management support for the product team.*

Furthermore, the need to clearly separate product discovery from product creation has been stressed in literature [1]. Such separation helps to clearly separate roles and responsibilities. For example, within UCD it is important to distinguish between the researcher and a prototyper [57]. Within ASD, the importance of clearly upfront specified roles and responsibilities have been suggested [55]. Given its importance, we also find it rooted in SCRUM, one example method of ASD. Therefore, we suggest (**DR12**): *A method for Software Product Discovery should clearly distinguish different roles and their responsibilities for the product team.*

Gaining a unified understanding of the product in planning (DR9), providing thorough guidance for designers and developers (DR10), ensuring management support (DR11) as well was specifying and separating team roles and responsibilities (DR12) are design requirements focusing on the team, i.e. all human resources involved. The last design principle **DP3** summarizes these team requirements: *A method of Software Product Discovery should guide a diverse team of specialists to develop a unified understanding of the product and its importance.*

6 The PDISC Method

Following the extraction of the methods and method fragments from prior literature, PDISC - a method for product discovery – is built. Starting with an initial idea generation (cf. [21, 39, 53]), each stakeholder can make suggestions in order to develop an idea pool. Interviews or focus groups help to gather ideas early on. Thereafter, an iterative process allows the execution of activities multiple times. Three main activities follow and can be executed in parallel. First, users are engaged and integrated into the discovery process [21, 31, 37, 40, 43, 54]. Marketers or developers use general or contextual interviews, task analysis or other practices to engage the user and collect requirement. Second, a central product vision document is created, so that a common

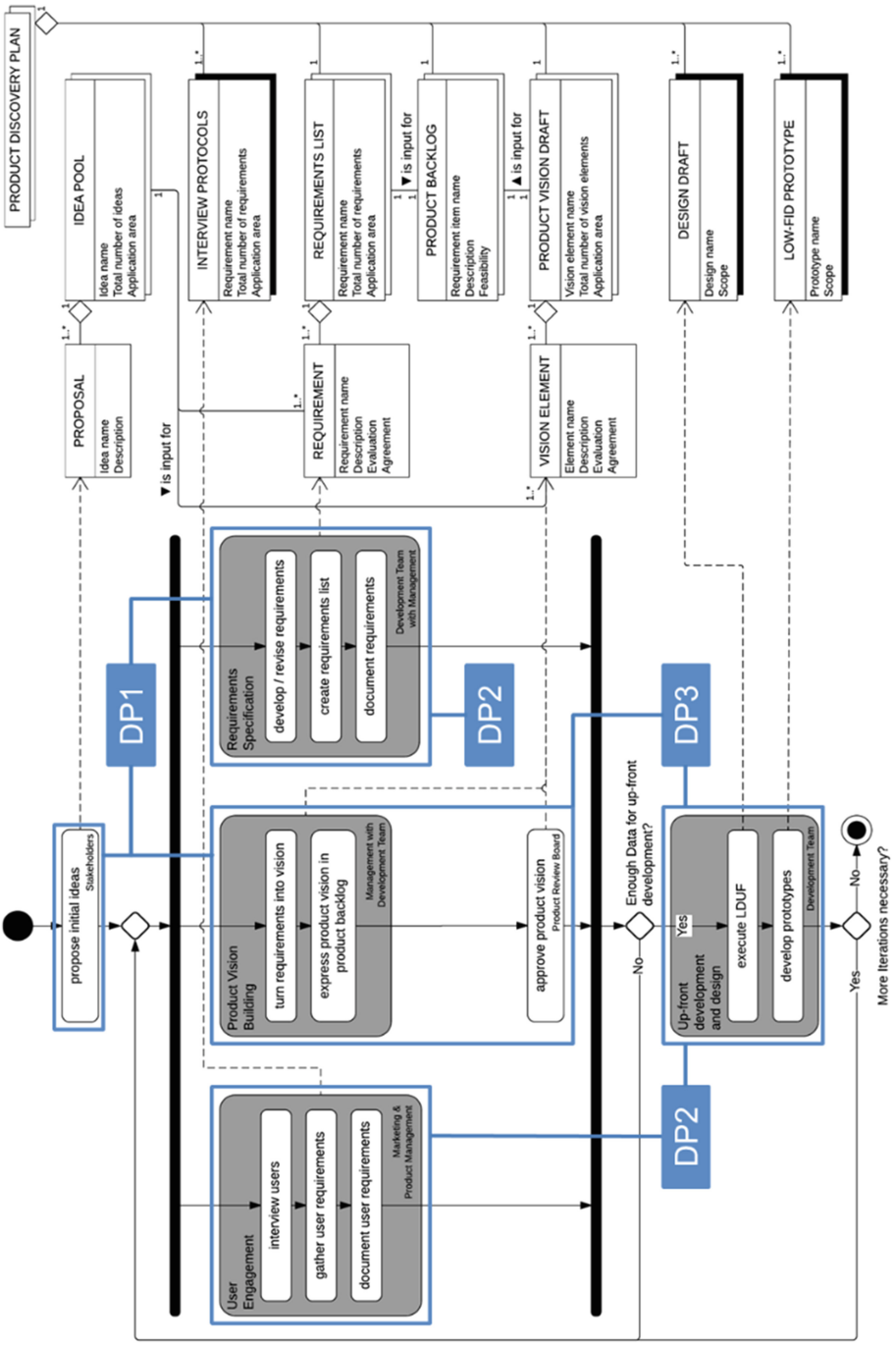

Fig. 3. PDISC, method for software product discovery. Described as a PDD with an overlay of the associated design principles related to the activities.

understanding is developed and documented for management approval [33, 37, 39, 54]. Management and the development team use prioritization techniques to select requirements and convert them into the product backlog. Third, requirements are generated, initially based on the idea pool [21, 31, 33, 37, 43]. In later iterations, such requirements are refined using the results from the user engagement. The documentation of user stories or scenarios can help the development team to communicate such requirements to management (Fig. 3).

The product vision draft and the requirements list form the product backlog [33, 39]. Until this point, the product backlog is the central element combining different activities and serving as input for any later phases. While this rather technical pool of functionality provides a valuable source for developers, different roles might need other types of documentation. The iterative cycle of the main activities ends when sufficient information for an initial design draft or low-fidelity prototype are available [31, 37, 43]. Design drafts are generated using sketching or mock-up applications. Early prototypes are created using paper. Later, tools help to create digital and interactive prototypes. However, the up-front development and design activities can only start after there are user inputs or requirements. However, once a design suggestion is available, feedback and a jump to the beginning of the iterative cycle are possible.

7 Conclusion

The study successfully derives design principles for a software product discovery method. Their implementation leads to the design of PDISC. Our theoretical contribution is the design principles, which categorizes product-, user- and team-related requirements. PDISC balances requirements for all three areas in order to deliver a viable, desirable, and feasible product vision. The product-related requirements help the organization to design a viable product. Implementing user-related requirements into the method assures that the product vision proposes a desirable product. In addition, team-related requirements of the method suggest the design of a feasible product. The design principles allow practitioners to challenge their own processes for comprehensiveness and completeness. While they may not implement all steps, depending on the size of the organization, PDISC helps practitioners to design a product that is viable, feasible and desirable. If a product falls short on any of these three dimensions, the product's success is at risk. Furthermore, the implementation and individual activities provide stepwise tutorial for creating a product vision. This is especially valuable for those organizations that yet have to define a software product discovery process for themselves.

References

1. Fox, D., Sillito, J., Maurer, F.: Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry. In: Agile 2008 Conference, pp. 63–72. IEEE Computer Society, Toronto (2008)
2. The Standish Group: Chaos Report, Las Vegas, NV, US (2014)

3. Brhel, M., Meth, H., Maedche, A., Werder, K.: Exploring principles of user-centered agile software development: a literature review. Inf. Softw. Technol. **61**, 163–181 (2015)
4. Barksdale, J.T., McCrickard, D.S.: Software product innovation in agile usability teams: an analytical framework of social capital, network governance, and usability knowledge management. Int. J. Agile Extreme Softw. Dev. **1**, 52 (2012)
5. da Silva, T.S., Silveira, M.S., de O. Melo, C., Parzianello, L.C.: Understanding the UX designer's role within agile teams. In: Marcus, A. (ed.) DUXU 2013, Part I. LNCS, vol. 8012, pp. 599–609. Springer, Heidelberg (2013)
6. da Silva, T.S., Silveira, M., Maurer, F.: Best practices for integrating user-centered design and agile software development. In: 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction, pp. 43–45. Brazilian Computer Society, Porto Alegre, Brazil (2011)
7. Ferreira, J., Boyland, J., Biddle, R.: Up-Front Interaction Design in Agile Development. In: Concas, G., Damiani, E., Scotto, M., Succi, G. (eds.) XP 2007. LNCS, vol. 4536, pp. 9–16. Springer, Heidelberg (2007)
8. da Silva, T.S., Martin, A., Maurer, F., Silveira, M.S.: User-centered design and agile methods: a systematic review. In: 2011 AGILE Conference, pp. 77–86. IEEE, Salt Lake City (2011)
9. Nwaka, S., Hudson, A.: Innovative lead discovery strategies for tropical diseases. Nat. Rev. Drug Discov. **5**, 941–955 (2006)
10. Cooper, R.G.: Winning at New Products: Creating Value Through Innovation. Basic Books, New York (2011)
11. Cagan, M.: Product Discovery. http://www.svpg.com/product-discovery
12. Freeman, P., Hart, D.: A science of design for software-intensive systems. Commun. ACM **47**, 19 (2004)
13. Boehm, B.W.: A spiral model of software development and enhancement. Computer (Long. Beach. Calif.) **21**, 61–72 (1988)
14. Sohaib, O., Khan, K.: Integrating usability engineering and agile software development: a literature review. In: 2010 International Conference on Computer Design and Applications, pp. V2-32–V2-38. IEEE, Qinhuangdao (2010)
15. Heikkilä, V.T., Paasivaara, M., Rautiainen, K., Lassenius, C., Toivola, T., Järvinen, J.: Operational release planning in large-scale Scrum with multiple stakeholders–A longitudinal case study at F-Secure Corporation. Inf. Softw. Technol. **57**, 116–140 (2015)
16. Kuusinen, K.: Improving UX work in scrum development: a three-year follow-up study in a company. In: Sauer, S., Bogdan, C., Forbrig, P., Bernhaupt, R., Winckler, M. (eds.) HCSE 2014. LNCS, vol. 8742, pp. 259–266. Springer, Heidelberg (2014)
17. Stevens, E.: Fuzzy front-end learning strategies: exploration of a high-tech company. Technovation **34**, 431–440 (2014)
18. Frishammar, J., Florén, H., Wincent, J.: Beyond managing uncertainty: insights from studying equivocality in the fuzzy front end of product and process innovation projects. IEEE Trans. Eng. Manag. **58**, 551–563 (2011)
19. Sperry, R., Jetter, A.: Theoretical framework for managing the front end of innovation under uncertainty. In: Portland International Conference on Management of Engineering & Technology, pp. 2021–2028. IEEE, Portland (2009)
20. Kakar, A., Carver, J.: Best practices for managing the fuzzy front-end of software development (SD): insights from a systematic review of new product development (NPD) literature. In: Proceedings of International Research Workshop on IT Project Management, p. 14., AISeL, Orlando, FL, US (2012)
21. Khurana, A., Rosenthal, S.R.: Towards holistic front ends in new product development. J. Prod. Innov. Manag. **15**, 57–74 (1998)

22. Kuechler, B., Vaishnavi, V.: On theory development in design science research: anatomy of a research project. Eur. J. Inf. Syst. **17**, 489–504 (2008)
23. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**, 45–77 (2007)
24. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-modelling based assembly techniques for situational method engineering. Inf. Syst. **24**, 209–228 (1999)
25. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Inf. Softw. Technol. **38**, 275–280 (1996)
26. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. Manag. Inf. Syst. Q. **28**, 75–105 (2004)
27. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (2007)
28. Brocke, J. vom, Simons, A., Niehaves, B., Reimer, K., Plattfaut, R., Cleven, A.: Reconstructing the giant: on the importance of rigour in documenting the literature search process. In: European Conference on Information Systems, p. 161. AISeL, Verona, IT (2009)
29. Boell, S.K., Cecez-Kecmanovic, D.: On being systematic in literature reviews in IS. J. Inf. Technol. **30**, 161–173 (2015)
30. Hildenbrand, T., Meyer, J.: Intertwining lean and design thinking: software product development from empathy to shipment. In: Maedche, A., Botzenhardt, A., Neer, L. (eds.) Software for People, pp. 217–237. Springer, Heidelberg (2012)
31. Hollis, B., Maiden, N.: Extending agile processes with creativity techniques. IEEE Softw. **30**, 78–84 (2013)
32. Kajko-Mattsson, M., Nyfjord, J.: A model of agile evolution and maintenance process. In: 42nd Hawaii International Conference on System Sciences, pp. 1–10. IEEE, Big Island (2009)
33. Nyfjord, J., Kajko-Mattsson, M.: Degree of agility in pre-implementation process phases. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 234–245. Springer, Heidelberg (2008)
34. Qumer, A., Henderson-Sellers, B.: Construction of an agile software product-enhancement process by using an agile software solution framework (ASSF) and situational method engineering. In: 31st Annual International Computer Software and Applications Conference, vol. 1, pp. 539–542. IEEE, Beijing (2007)
35. Sarpong, D., Maclean, M.: Mobilising differential visions for new product innovation. Technovation **32**, 694–702 (2012)
36. Sibghatullah, M., Hussain, S., Hussain, S.: An approach to effective product development life cycle. In: International Conference on Emerging Technologies, pp. 719–726. IEEE, Peshawar (2006)
37. Tessarolo, P.: Is integration enough for fast product development? An empirical investigation of the contextual effects of product vision. J. Prod. Innov. Manag. **24**, 69–82 (2007)
38. Vanhanen, J., Itkonen, J., Sulonen, P.: Improving the interface between business and product development using agile practices and the cycles of control framework. In: Proceedings of the Agile Development Conference, pp. 71–80. IEEE, Salt Lake City (2003)
39. Cloyd, M.H.: Designing user-centered web applications in web time. IEEE Softw. **18**, 62–69 (2001)
40. Patton, J.: Hitting the target: adding interaction design to agile software development. In: OOPSLA 2002 Practitioners Reports, Salt Lake City, p. 7 (2002)

41. Ben Rejeb, H., Boly, V., Morel-Guimaraes, L.: A new methodology based on Kano model for the evaluation of a new product acceptability during the front-end phases. In: 32nd Annual IEEE International Computer Software and Applications Conference, pp. 619–624. IEEE, Turku (2008)
42. Salah, D., Paige, R., Cairns, P.: A practitioner perspective on integrating agile and user centred design. In: 28th International BCS Human Computer Interaction Conference, Southport, UK, pp. 100–109 (2014)
43. Adikari, S., McDonald, C., Campbell, J.: Little design up-front: a design science approach to integrating usability into Agile requirements engineering. In: Jacko, J.A. (ed.) HCI International 2009, Part I. LNCS, vol. 5610, pp. 549–558. Springer, Heidelberg (2009)
44. Miller, L.: Case study of customer input for a successful product. In: Agile Conference, pp. 225–234. IEEE, Denver (2005)
45. Salah, D., Paige, R.F., Cairns, P.: A systematic literature review for agile development processes and user centred design integration. In: 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–10. ACM Press, New York (2014)
46. Knoll, S.W., Horton, G.: The impact of stimuli characteristics on the ideation process: an evaluation of the change of perspective "Analogy." In: 44th Hawaii International Conference on System Sciences, pp. 1–10. IEEE, Kauai (2011)
47. Oliveira, M.G., Rozenfeld, H.: Integrating technology roadmapping and portfolio management at the front-end of new product development. Technol. Forecast. Soc. Change **77**, 1339–1354 (2010)
48. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51**, 915–929 (2015)
49. Loniewski, G., Armesto, A., Insfran, E.: An architecture-oriented model-driven requirements engineering approach. In: Model-Driven Requirements Engineering Workshop, pp. 31–38. IEEE, Trento (2011)
50. Salvador, C., Nakasone, A., Pow-Sang, J.A.: A systematic review of usability techniques in agile methodologies. In: 7th Euro American Conference on Telematics and Information Systems, pp. 1–6. ACM Press, New York (2014)
51. Ferre, X., Medinilla, N.: How a human-centered approach impacts software development. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4550, pp. 68–77. Springer, Heidelberg (2007)
52. Menor, L.J., Tatikonda, M.V., Sampson, S.E.: New service development: areas for exploitation and exploration. J. Oper. Manag. **20**, 135–157 (2002)
53. Sy, D.: Adapting usability investigations for agile user-centered design. J. Usability Stud. **2**, 112–132 (2007)
54. Jain, R., Suman, U.: A systematic literature review on global software development life cycle. ACM SIGSOFT Softw. Eng. Notes **40**, 1–14 (2015)
55. Ebert, C.: Understanding the product life cycle: four key requirements engineering techniques. IEEE Softw. **23**, 19–25 (2006)
56. Williams, H., Ferguson, A.: The UCD perspective: before and after agile. In: Agile Conference, pp. 285–290. IEEE, Washington, D.C. (2007)
57. Liskin, O.: How artifacts support and impede requirements communication. In: Fricker, S. A., Schneider, K. (eds.) REFSQ 2015. LNCS, vol. 9013, pp. 132–147. Springer, Heidelberg (2015)

58. Gaubinger, K., Rabl, M.: Structuring the front end of innovation. In: Gassmann, O., Schweitzer, F. (eds.) Management of the Fuzzy Front End of Innovation, pp. 15–30. Springer International Publishing, Cham (2014)
59. Gamble, R.F., Hale, M.L.: Assessing individual performance in Agile undergraduate software engineering teams. In: IEEE Frontiers in Education Conference, pp. 1678–1684. IEEE, Oklahoma City (2013)