Software Product Manager: A mechanism to manage software products in small and medium ISVs

Rudy Katchow [1], Inge van de Weerd [1], Sjaak Brinkkemper [1], Andy Rooswinkel [2]

[1] Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{r.katchow , i.vandeweerd, s.brinkkemper}@cs.uu.nl
[2] Servoy, Amersfoort, The Netherlands
arooswinkel@servoy.com

Abstract. In this paper, we present SP Manager as an innovative tool for managing software products in small and medium independent software vendors (ISVs). This tool incorporates the operational software product management (SPM) processes focused on requirements management and release planning. By using situational method engineering techniques, the tool is easy to adapt to a specific company context. In addition, by making it possible to integrate the tool with the development platform, the tool is easy to deploy and adopt. The expert validation of this tool indicates that the included development concepts, such as the integration of SPM with system defect management, provide additional advantages to other SPM tools in the market.

Keywords: Software product management, small and medium ISVs, situational method engineering, system defect management, requirement management, release planning.

1 Software product management in small and medium ISVs

Small and medium enterprises (SMEs) developing software are dynamic, innovative, and efficient enterprises with fast growth potential; they are considered as the motors that agitate the industrial growth [1]. There is an evident need for SMEs developing software to improve their software product management (SPM) processes with the aim to react on the demands of the market (high quality software, fast development cycle, low costs, and advanced features) and in order to face the increasing global competition [2] [3].

Nikula et al. [4] realized that most small and medium independent software vendors (ISVs) have poor practices to conduct their requirements engineering processes, despite the solid academic knowledge that is available in this domain. Kamsties et al. [1] added that small and medium ISVs are characterized by being sensitive to expenses, and opting for mature results being transferred in a short time period. This fact is also indicated in a recent survey among SMEs developing software [5], which shows that most of these SMEs are familiar with the different modeling methods and tools for the requirements engineering and analysis phases of software development. However, the main constrains that are limiting these companies from improving their software processes are a lack of resources in terms of time, training expenses, and qualified staff.

The gap between the academic world and the practitioners' world could be reduced, according to Nikula et al. [4], in the form of training, examples, templates, and adoption of standardized tools. They also found that in managing requirements, many small and medium ISVs consider adopting specific tools to be a costly approach. Accordingly, they tend to use general-purpose text processing and spreadsheet solutions.

The presented overview indicates that small and medium ISVs need a mechanism to acquire the wide academic knowledge in SPM rapidly and employ this knowledge to conduct their SPM processes in a systematic approach that can easily be adapted to their specific situations. This leads to the following research question:

How to develop an innovative mechanism to manage software products for small and medium ISVs?

The answer to this question is attained by developing of an innovative tool for managing software products in small and medium ISVs. The concepts used in developing this tool include the necessary aspects for small and medium ISVs that are not covered entirely by the other SPM products in the market. We call this tool *Software Product Manager* (SP Manager).

This paper is organized as follows. In section 2, we give an outline about the research method that is applied. The concepts that are used to develop SP Manager are described in section 3. Then, in section 4, we provide an overview on the development approach of this tool. In section 5, we describe our validation method and then analyze the obtained results. Section 6 presents the related work, and we conclude this paper in section 7 with a summary and directions for future work.

2 Research method

In conducting this research, we follow the design research method [6]. Design research involves the creation, analysis of use, and evaluation of designed artifacts to understand, explain, and to improve the behavior of aspects of Information Systems [6]. The goal of design science is 'utility' that is achieved by meeting business needs through building and evaluating the designed artifact. This approach differs but complements the traditional behavioral science research that aims to seek the 'truth' through developing and justifying theories that explain or predict phenomena related to business needs [7].

It is necessary to distinguish design research from routine design that applies the existing knowledge of a problem space to solve business problems. Design research contributes to the business and scientific realm by finding new artifacts or methods to address business problems using the existing knowledge base, which makes the solution 'innovative'. Furthermore, it can also result in 'creative' solutions if the knowledge needed to address a problem is not available in a form directly applicable to the problem [8].

In Figure 1, we depicted the information systems research framework as proposed by Hevner et al. [7]. The business needs that initiate our research question are originated from the SPM environment. Addressing these needs is achieved through building and evaluating of an artifact or theory (in our research a tool *SP Manager*). The tool is then optimized in an iterative assessment and refinement process. The knowledge base provides the foundations (such as SPM literature and situational method engineering theory) and research methodologies (such as design research approaches, interview techniques) that are applied in our research in an appropriate manner, hence ensuring research rigor. The result of this design research, SP Manager, contributes to the business environment by satisfying business needs from one side and adding to the contents of the knowledge base for future research and practices on the other side.

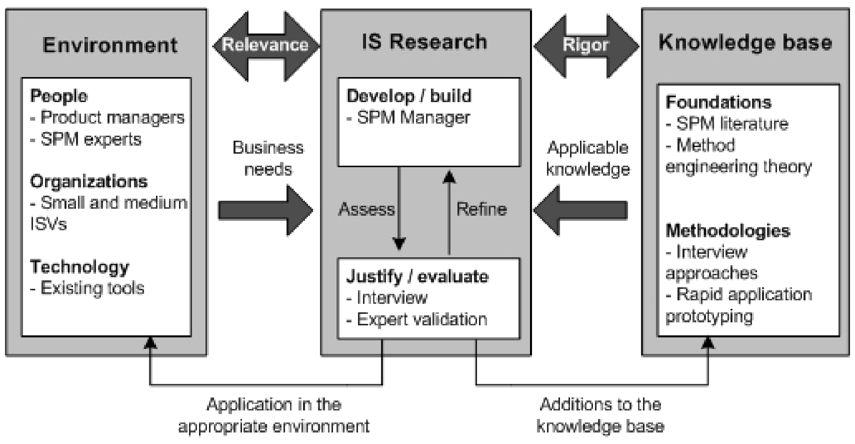


Figure 1: Research visualized in the Information Systems Research Framework [7]

3 Development concepts

3.1 The reference framework for software product management

In small and medium ISVs, it is common that some software processes, such as SPM processes, have not been defined and implemented at all. Moreover, small ISVs sometimes find it difficult to finance and carry out large and expensive process improvement efforts. In this situation, small and medium ISVs can greatly benefit from benchmarking the processes defined in similar ISVs [3].

In order to assure that the processes carried out by our proposed SPM tool covers all the necessary SPM processes for small and medium ISVs, we base our development of SP Manager on the reference framework for SPM. This framework that is introduced by Weerd et al. [9] provides the key processes in SPM that should be conducted in each ISV as well as SPM stakeholders and their relations. The framework described four main process areas within SPM: requirements management, release planning, product roadmapping and portfolio management. We focus only on requirements management and release planning, since we limit our research to the operational processes. Portfolio management that specifies in more detail the projects needed to fulfill the strategic goals of an ISV [10], and product roadmapping are not covered in this research.

3.2 Situational method engineering

In research by Nooteboom et al. [11] about theory of transaction cost economics for small enterprises, it is concluded that (from a user perspective) using standardized products results in some potential advantages, such lower costs of switching from one product to another and less investments in knowledge about the product. However, there are also some potential disadvantages, for example less tailoring of eccentric processes for SMEs, and less competition to satisfy the demands of the users.

Despite the fact that most ISVs prefer a standard product to perform their SPM processes, developing any general solution is hardly applicable [4] [12] since the standard solution should be adapted to the situation of each ISV prior to apply it [13]. Therefore, we pursue the concept of situational method engineering. A ***situational method*** is an information systems development method tuned to the situation factors of the system [14]. For each situation (in our research ISV), a different method is built or an existing method is tuned to that situation. ***Situational method engineering*** is based on selecting the optimal method fragments from a method base (repository of method fragments) induced by the characterization of an ISV [15]. The accumulated experience from applying the selected method fragments can help in refining and optimizing their choice in future implementations.

An ISV can be characterized by describing its ***situational factors***, which are defined as a combination of circumstances at a given moment in a given ISV [16]. A large number of situational factors have been investigated in earlier research. Harmsen et al. [17] categorized situational factors into three groups: environment, project organization, and other situation factors. Slooten & Hodes [18] used another categorization approach, in which a companies situation was described by contingency factors and constraint factors.

In the SPM domain, Bekkers et al. [19] identified, through collected literature and empirical cases studies, the essential situational factors influencing the selection of method fragments for SPM processes. In their research, they also identified the situational factors' weights that show how influential specific situational factors are to each process in the reference framework for SPM. We use the list of situational factors identified by Bekkers et al. [19] as a guideline to select the optimal method fragments for SPM processes according to the situational factors of each ISV.

3.3 Integrating SPM tools with development platforms

The concept of building integrated environments has evolved over the last three decades and resulted in many prototypes of Software Development Environments [SDES], which are highly integrated set of tools supporting the complete software development process. The ultimate goal of SDES is to improve the product's quality, to support reusability, and to free developers from routine work [20].

According to a Forrester report [21], integrating requirements management tools with development platforms gives product managers the opportunity to monitor and control the work progress of the development team, hence supporting decision making and ensuring that the development team delivers the required functionality on time and under budget.

Repenning et al. [22] have also realized the necessity of efficient collaboration between development team who are technologically savvy and product managers who are experts in their product's domain but might not adequately comprehend technological limitations or opportunities.

Integrating requirements management tools with development platform are already implemented in several applications. DOORs, Requisite Pro, and Cradle [23] are examples of control and management tools that provide support for both product managers and development team during the entire development lifecycle of software products [24]. Mingle and Cruise [25] integrate release planning with development platforms and provides product managers as well as the development team with an advanced visual interface and control over the developed products.

In designing SP Manager, we propose a simplified approach to integrate this tool with development platforms through a simple module or plug-in. This module is dedicated to the development team to provide them with an overview of the assigned tasks that need to be implemented for a given release. This module makes the collaboration between product managers and development team more efficient by allowing product managers to monitor the development status and ensure that the releases are achieved within the specified time and budget. It also gives the development team the possibility to request scope changes and check the status of their requests.

3.4 Integrating system defect management with requirements management and release planning

There is distinction between a ***requirement*** that is defined as a feature or behavior of the system that is desired by one or more stakeholders [26] and a ***system defect (bug)*** that is fault in existing features or functionalities that disrupt the system from functioning in the normal behavior. For small and medium ISVs, there are however no clear boundaries between requirements management and managing system defects. Therefore, it is necessary to integrate system defect management in SP Manager to provide one solution to all reported issues (an issue is an umbrella term that we use in this research to refer to both requirements and system defects).

In integrating system defect management with requirements management and release planning, we pursue the best practices as outlined by ITIL [27]. When managing system defects, it is necessary to distinguish between two concepts: incidents and problems. An ***incident*** is defined as an unplanned interruption to an IT service, reduction in the quality of an IT service, or failure of a configuration item that has not yet impacted the IT service. A ***problem***, on the other hand, is defined as the cause of one or more incidents [27]. By resolving the route causes of a problem, and including the solution implementation in the release planning process, recurring incidents can be eliminated.

The proposed conceptual model for the integration of system defect management with requirements management and release planning is shown in Figure 2.

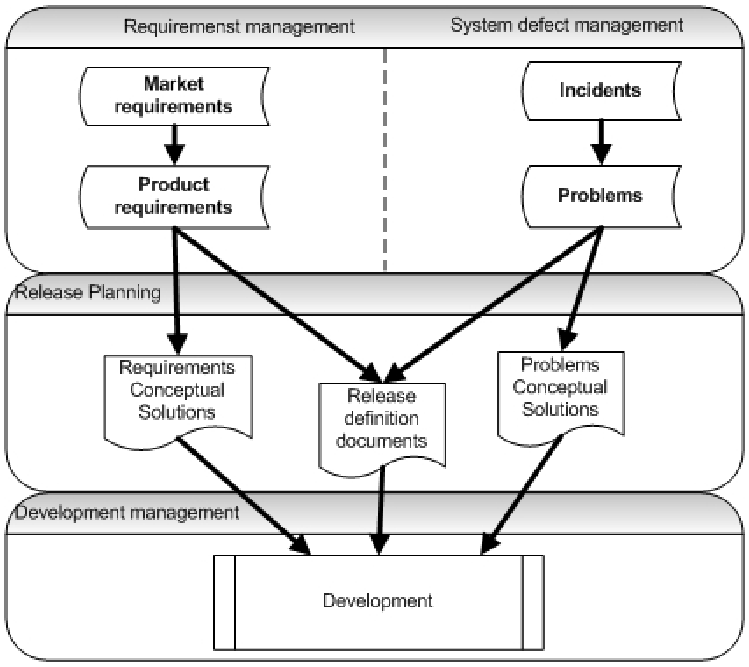


Figure 2: Conceptual model for integrating system defect management with requirements management and release planning

Once a problem is identified from one or more incidents, it can be included in the release planning processes together with the selected product requirements (PRs) or it can be selected separately as hot-fix release. For both product requirements and problems, conceptual solutions should be written, containing the proposed functional solutions that will help the development team in implementing the reported product requirements and problems.

3.5 Managing customized product branches

A software product ISV is a company that provides packaged software products for a target market rather than developing customized software products to specific customers. This is in opposite of a software service enterprise that develops and sell ***customer customized solutions*** (bespoke) based on the order of specific customers [28]. In practice, ISVs rarely develop either software products or customized software but a compilation of both, where one or the other model is dominant [29].

According to Brereton [30], producing market-driven packaged products favors large ISVs that control the evolution of their products compared to small and medium ISVs that come under the influence of their large customers to implement their specific requirements.

In research on development strategies, Kamsties et al. [1] have identified that small and medium ISVs either start directly with a market-driven development strategy or they generalize a solution from one initial customer to the needs of other similar customers. Furthermore, they found that SMEs developing software must be adaptable to customers' needs. The degree of adaptability varies from user-configurable issues, such as the appearance of the graphical user interface, to vendor-configured issues, such as specific functionality required by individual customers.

In Figure 3, we propose an approach to manage software products and customized product releases in SP manager.

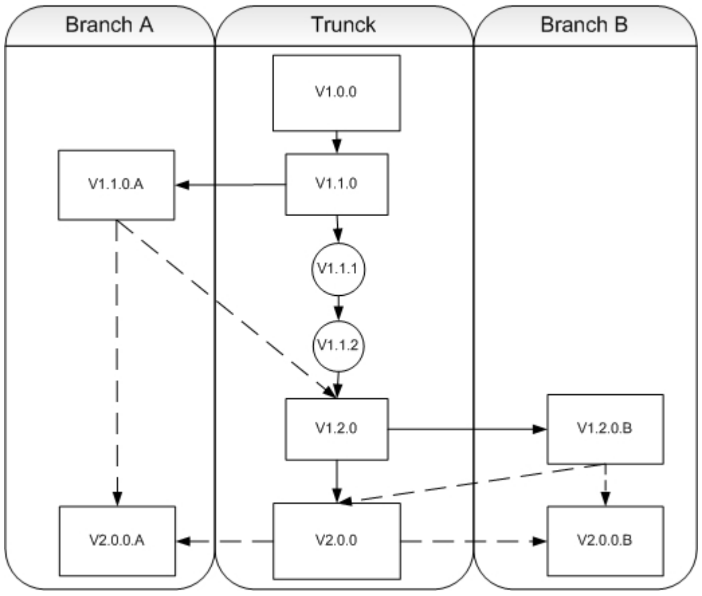


Figure 3: An approach to manage software products and customized product releases

The customized release for *Customer A* in release *V1.1.0.A* includes requirements specific to this customer. Not all these requirements are merged back to the main product line (trunk) in the subsequent release (this might be due to conflict between different requirements or specificity of certain requirements that are not necessarily applicable to other customers). In order for *Customer A* to maintain its customized release, it should be updated separately with new release *V2.0.0.A* that includes new requirements from the main release line *V2.0.0* and the specific requirements from the previous release branch. The releases symbolized in a circle represent hot-fix releases that include solutions to system problems. These hot fixes are merged in the subsequent release(s).

3.6 Managing product requirements granularities and their dependencies

Small and medium ISVs can have hundreds or thousands of detailed requirements, which are too many to classify analytically and consistently. Depending on the situation of each ISV, the abstraction level of PRs can be at the use case level, feature level, or the detailed functional requirement level [31].

The list of PRs selected for a release should reflect a coherent extension of the product. Too many small PRs make this list difficult to be managed while too large PRs deprive an adequate insight of their content and hinder the effective communication between development team and product manager [32]. In order to assure coherency between PRs in term of their size, Natt och Dag et al. [32] suggest that similar effort in man-days required for development can be used as bases for defining the level of abstraction of PRs.

The granularity concept is not applied to market requirements (MRs), since the customers can express their wishes at different levels of granularity [32]. It is possible to link large MR to many different PRs, also when MRs are small then we can bundle them to be linked to one PR. For example, MRs that deal with small user interface improvement requests can be linked to one PR that manages graphical user interface (GUI) improvements. We suggest following these guidelines for abstracting and bundling PRs during the implementation of SP Manager. In Figure 4, we outline the relationship between MRs and PRs.

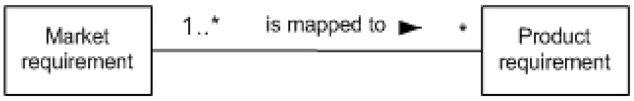


Figure 4: The relationship between market requirements and product requirements

The interdependency between PRs is also essential when refining and selecting them to be released. In an ideal situation, where there are no dependencies between PRs, release planning is just a matter of prioritizing them and selecting a number of top priority PRs depending on the available resources and the delivery date. However, in practical situation only small percentage of PRs are independent [31].

In designing SP Manager, we follow the classification scheme for dependencies [33] that are organized as functional-based dependencies (implication, mutual, and exclusion), revenue-based dependencies that occur when combinations of PRs increase or decrease the revenue value, and cost-based dependencies that occurs when combinations of PRs increase or decrease the required amount of resources [34].

4 Development of SP Manager

We use ***Rapid Application Development*** (RAD) method for developing SP Manager. RAD method, initiated by James Martin [35], involves the construction of prototypes in successive iteration of development to refine and improve these prototypes to production [36]. This method is typical for small scale projects and of short duration. RAD is also characterized by small size development team with experience of the used technologies and an actively involved management that is vital to mitigate the risks of lengthened development cycles [37].

Approaches to RAD requires support tools for rapid changes that are represented by a combination of fourth generation languages, GUI builders, database management systems (DBMS), code generators, and computer-aided software engineering (CASE) tools. The combinations of most of these tools are included in Servoy 4.0 development platform [38] that we use in developing SP Manager.

The functional diagram of SP Manager is shown in Figure 5, where a scenario for a release that includes PRs and 'minor' problems is demonstrated. All reported issues are first gathered and classified. During the system defect identification process, incidents and problems are recognized. Incidents are usually handled by 'service and support' that conduct support procedures to return the service to the normal situation [39]. Service and support can also detect problems that will be reported as new issues. The identified problems are diagnosed to indicate whether they can be solved by workaround (a temporary way to overcome system defects) or through permanent solutions. Problems are then prioritized, where 'minor' problems are recorded to be included in the next release(s), and for 'major' problems, which involve critical problems that need to be solved immediately, a hot-fix release is prepared.

In the scenario shown in Figure 5, sequence (a) represents the functional flow to select minor problems to be included in a given release, while sequence (b) represents the functional flow to select PRs. These two sequences are then merged into sequence (c) where the selected PRs and 'minor' problems are included in the release definition documents. The percentage of PRs and problems in a release is a trade-off decision between stabilizing the product in the subsequent release, and implementing new functionalities.

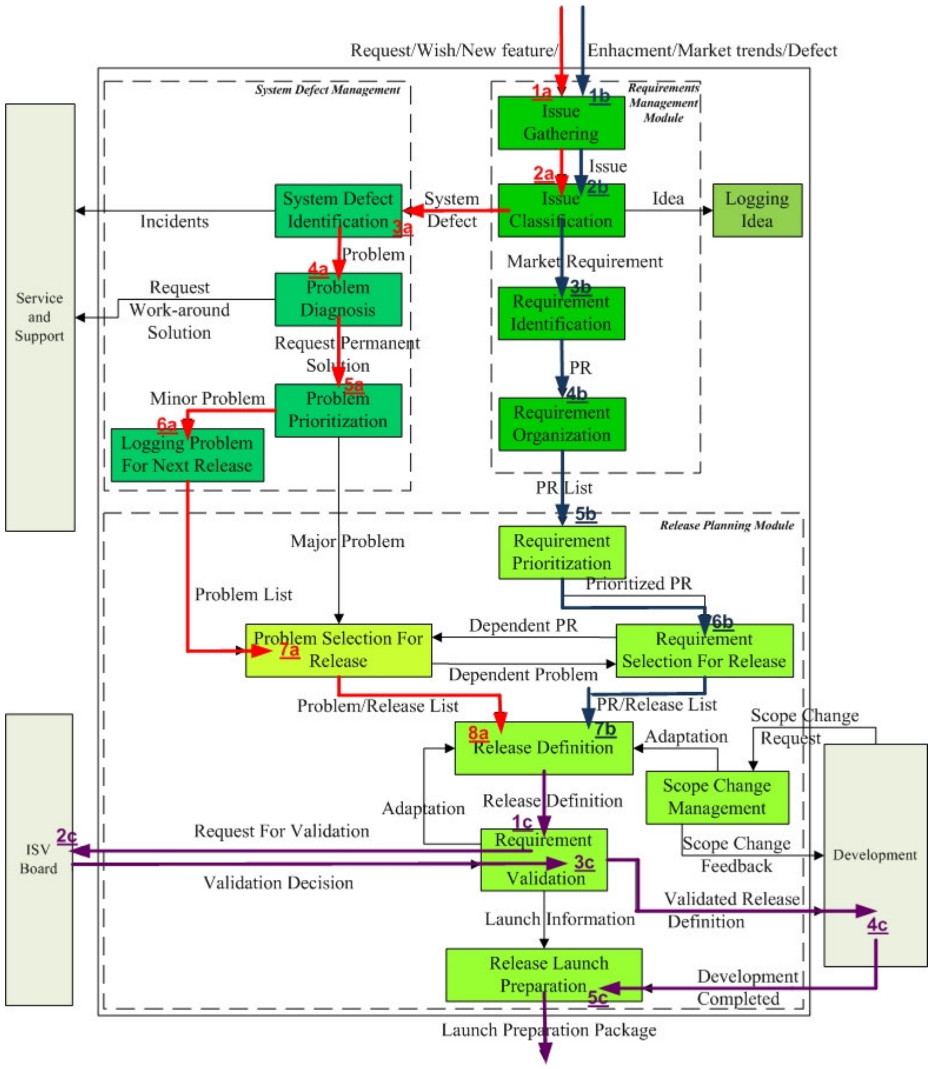
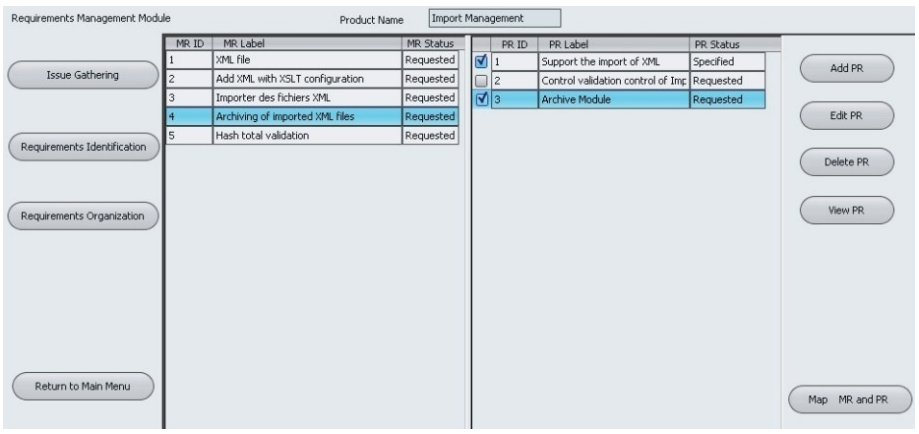


Figure 5: Scenario for a release that include both PRs and 'minor' problems

Figure 6 shows a screenshot of a user interface for SP Manager, showing the requirements identification screen, in which MRs are linked to PRs. Other screens in the requirements management module are issue gathering, issue classification and requirements organization. The other two modules are the system defect module and the release planning module. In Figure 5, these modules are further specified.


Requirements Management Module
Issue Gathering
Requirements Identification
Requirements Organization
Return to Main Menu
Add PR
Edit PR
Delete PR
View PR
Map MR and PR

Product Name: Import Management

MR ID	MR Label	MR Status
1	XML file	Requested
2	Add XML with XSLT configuration	Requested
3	Importer des fichiers XML	Requested
4	Archiving of imported XML files	Requested
5	Hash total validation	Requested

	PR ID	PR Label	PR Status
✓	1	Support the import of XML	Specified
☐	2	Control validation control of Imp	Requested
✓	3	Archive Module	Requested

Figure 6: Screenshot of a user interface for SP Manager

5 Validation Stage

We validate SP Manager through involving a panel of SPM experts in examining the development concepts introduced in SP Manager. The SPM expert's judgment indicates whether this tool meets the objectives that are defined for this research.

5.1 Validation approach

Relying on expert opinion as a mean of evaluation is recognized by many researcher. Rosqvist et al. [40] suggested a method to use expert judgment for the evaluation of software quality. Kitchenham et al. [41] have indicated that there is increasing use of expert opinion based models such as Bayesian Belief Networks and System Dynamic Models in validating software engineering problems.

Expert validity is a form of content validity, which is demonstrated by asking experts to review the content of the designed artifacts. Expert assessment may be quantitative, qualitative or simultaneous triangulation of both quantitative and qualitative methods. Quantitative Expert assessment enables the determination of the content validity index [42]. In this research, we use qualitative approach to validate SP Manager. Qualitative data can be used for exploring content validity of an artifact allowing experts to provide suggestions to improvement or supply new ideas [42]. Qualitative methods would also counter the subjectivity and interpretive relativism that are seen as both advantage and liability in qualitative research [43].

5.2 Expert panel and validation instrument

In our research, we define an expert as a person who has published widely in recognized scientific journals in the field of SPM, or has practical experience of SPM holding position as a product manager for several years [44]. These experts are presumably able to provide an informed, objective, and unbiased opinion as well as suggestions about the developed artifacts [43].

Regarding the size of the expert panel, it should be at least five experts according to Lynn [45]. Hyrkäs et al. [42] suggest that ten experts would provide a reliable determination of an artifact's content validity. Therefore, we have selected ten SPM experts (5 product managers and 5 SPM experts from academia) to participate in validating SP Manager.

The data collection method is based on conducting an open questions questionnaire during our interview with selected SPM experts. The questionnaire is composed of 27 questions. For each development concept, we propose several questions to our interviewees to adhere their opinions as well as identifying the strengths and weaknesses of the suggested approaches for these development concepts. The final questions are about the general impression on SP Manager.

5.3 Analysis of results

5.3.1 The reference framework for software product management

There is strong agreement between our interviewees that the processes presented in the reference framework for SPM in requirements management and release planning phases [46] cover the essential processes to manage software products. Our expert panel agreed that the same processes outlined in this reference framework holds also for small and medium ISVs, although in these ISVs one person can have different roles in conducting these processes.

However, some of our SPM experts indicated that other processes are required to share knowledge between different stakeholders. For example it is necessary to communicate with the customers or other software ecosystem stakeholders before making decision about selecting the PRs to be released.

5.3.2 Situational method engineering

There is a positive consensus amongst our SPM experts about the approach of providing one tool with different methods that can be selected according to the situation of each ISV. This approach provides flexibility to tune the available method fragments to the situation of small and medium ISVs. One expert warned though that the easiness of change might decrease the stability of managing software products.

Using situational method engineering approach allows ISVs to add new methods based on their best practices and to grow in process maturity by replacing their methods with mature ones without the need to integrate or to migrate to other tools. However, the infrastructural design should be left open in order assure the contingency and integration of method fragments.

A disadvantage of situational method engineering is the difficultly to link method fragments to situational factors and to adapt or change the methods dynamically during conducting them. One expert argued that the advocators of situational method engineering approach believe that they can solve all the problems in the world with this approach without taking into consideration the complexity of the real world.

5.3.3 SPM situational factors

While some of the expert panel members support the idea that most of the knowledge needed to choose method fragments are in the list of SPM factors suggested by Bekkers et al. [19], other members believed that these factors are subjective representation of the ISVs and lack the human style of an ISV such as emotions, skills, and culture. These SPM factors can serve as a starting point to select proper method fragments, but there is still a need for expert's gust to validate this selection. One expert explained that it might be sensitive to evaluate human factors such as the efficiency of product managers or the relationship between development teams. Another expert added that the identification of these situational factors can be different depending on the judgment of human expert who identify them.

5.3.4 Integrate SPM tools with development platforms

When asking our interviewees about the advantages of integrating SPM tools with development platforms, they believed that this integration can improve tracking and tracing, provide the necessary control on the assigned tasks to the development team, allow for central management, and improve the share of knowledge between product managers and development team. The disadvantage of this integration as perceived by our SPM experts is that it forces the developers to work based on certain routine, which make them not willing to cooperate with the product managers.

We also asked the SPM experts whether this integration would accelerate the development lifecycle, 70% of our experts opposed this idea. Yet the experts believe that this integration can enhance the interaction between product managers and development team and also improve product's quality.

5.3.5 Integrating system defect management with requirements management and release planning

There is a total agreement between the interviewed experts that managing system defect differs from managing requirements. System defect necessitate more technical explanation than requirements, they are also more time critical. System defects can be solved with workarounds and are categorized as technical defect in the source code or defect in requirement's definition. One expert considers developing PRs as a matter of choice, while solving system defect is sort of an obligation defined usually by the maintenance agreements. Another expert added that keeping system defect unsolved affect more severely the image of an ISV than keeping requirements unimplemented.

About our suggested conceptual model in Figure 2, there is a 90% support amongst our interviews to this model. One expert suggested that for system defects, it should be possible to bypass the define release document process and interact with development team in agile way depending on the severity of system defects.

5.3.6 Managing customized product branches

The members of the expert panel agreed that one of the unavoidable challenges that small and medium ISVs are facing is the pressure held by their large customers to release customized versions of their products. An ISV's financial situation is usually the main factor that forces an ISV to accept this approach. One expert said this is strategic choice between those who seek short and long term advantages.

Managing the combination of software product and customized product releases differs from one ISV to another. While some ISVs permit this approach by allowing the customers to raise (buy) the priority of their requirements at the condition that they should be integrated back to other customers in the next releases, other ISVs do not hold any responsibility regarding the maintenance and upgrade of the customized releases and usually outsource this to a sister ISV or a third partner. One expert added that some ISVs manage customized branches separately as separate products.

About our suggested approach in managing software product and customized product releases in Figure 3, our experts were not enthusiast with a lower support of 20%. This approach is good in theory when the products are properly designed and merged but in practice there are many interactions and conflicts between PRs.

5.3.7 Managing product requirements granularities and their dependencies

The main factors recognized by our SPM experts to define the granularity level of PRs are the time needed to implement them, the number of PRs in a defined release, functionality of PRs, and the development philosophy.

We also asked our interviewees to identify the types of dependencies that coexist between PRs. The identified dependencies are functional (include, partially include, mutual, exclude), cost, value, and parent-child relationship. One expert thinks that it is necessary to identify the sequence in which these PRs would be implemented and suggests developing the high risk PRs first to discover their effect on other PRs.

About the difficultly to identify dependencies between PRs, our experts recognized that the lack of technical knowledge is the main reason to limit product managers from identifying these dependencies. Also product managers could have different perspectives when identifying these dependencies based on their experience.

5.3.8 Overall impression

According to our SPM experts, about 60 to 70% of small and medium ISVs do not apply a structured and formal way of conducing SPM. The main reasons are unawareness of existing SPM tools in the market, the cost of the available SPM tools, the administrative tasks that come with SPM that are time consuming for small ISVs, and the belief that, small ISVs in particular, do not need SPM tools and can use general-purpose spreadsheet solutions.

There is positive consensus between the experts that combining the previously mentioned development concepts into SP Manager provides advantages over the other SPM products in the market. One SPM expert indicated that most of these concepts have solid bases since they are validated scientifically. Another advantage of this tool is the rapidness and low cost of developing new method fragments.

When considering further development of SP Manager, the panel of experts advises that more processes should be included to cover the marketing and development perspectives of product management. One expert suggested expanding SP Manager to be an ERP for small and medium ISVs. The experts also suggest including the concept of managing parallel releases. This means that a PR can have different statuses according to its different releases. Another point that is raised by our experts is the possibility to adapt the statuses workflow.

6 Related work

Most of the research on SPM regarding packaged products focuses in particular on large ISVs, despite the fact that most software is produced by small and medium ISVs [47]. In this section, we present the main research groups and the individual work that study SPM and then we outline some of the tools that manage software products.

The product software research group at the Information and Computing Science faculty of Utrecht University has performed extensive research in the field of software products and SPM. In this context we mention the work of Xu and Brinkkemper [48] in defining the core concept of software product, and Weerd et al. [46] who developed the reference framework for SPM.

The Software Engineering Research Group (SERG) at Lund University is active in conducting research on analysis and improvement of software processes in general and requirements engineering processes in particular, e.g. [49]. SoberIT is another research group that is part of Helsinki University of Technology, which performs research related to software business and the management aspects of software development. One of the largest projects in SPM field is REAIMS that concentrates on requirements process improvement by introducing a best practice based approach to assess requirements engineering processes at different ISVs [50].

Regarding the study of SPM in small and medium ISVs, we refer to the work of Nikula et al. [4] and Kamsties et al. [1] who have surveyed requirements engineering processes in SMEs developing software. In addition, Kilpi [3] has studied improving SPM processes in SMEs developing software.

Concerning SPM tools, Hoek et al. [51] have presented a tool for managing software release processes, and Ruhe & Saliu [52] have introduced a hybrid planning approach that integrate the strength of computational intelligence in conducting release processes with the knowledge of SPM experts. Commercially there are many tools available in the market that are targeted mainly at large ISVs, example of these tools are requirements management tools such as CaliberRM (Borland), Telelogic DOORS (IBM), Cradle (3SL), and Rational Requisite Pro (IBM) [23][53], and tools that focus on bug tracking are for example JIRA (Atlassian) [54], Lighthouse [55]. Other tools that support some processes of SPM are Telelogic Focal Point (IBM) that concentrates on prioritization of requirements [56].

7 Conclusion, discussion and future research

In this research, we have developed an innovative tool (SP Manager) for managing software products in small and medium ISVs. This tool that summarizes the body of knowledge in SPM and presents it as an easy to adapt and deploy tool. The development concepts included in SP Manager provide, according to a group of selected SPM experts, additional advantages to the other SPM tools in the market.

We have also shown how our selected group of SPM experts validated SP Manager. Their responses to our validation instrument highlight the potential strengths and weaknesses of the concepts that are used in developing SP Manager. The general attitude of the experts is supportive towards these development concepts. In addition, they indicate that this tool provides additional advantages for small and medium ISVs by including concepts that are not covered entirely by other SPM tools. Regarding the suggested approach to implement these concepts in SP Manager, there is positive consensus amongst the experts, with the exception of the approach to manage the combination of customized and software product releases. However, we are also aware that having designed the validation instrument and having chosen the expert panel's members ourselves may result in some bias in the evaluation.

The validation results are helpful in guiding us to further development of SP Manager. The research community and the software industry can also gain from the attitudes of the experts towards the development concepts in SP Manager. Further development of SP Manager can concentrate on the identified weaknesses during the validation of this tool, and extend this tool to cover the product roadmapping and product portfolio processes of SPM. Future research can focus on the long term impact of introducing SP Manager in small and medium ISVs, and whether the included development concepts in SP Manager are also suitable for large ISVs.

References

[1] Kamsties, E., Hörmann, K., & Schlich, M. (1998). Requirements engineering in small and medium enterprises. *Requirements Engineering , 3* (2), 84-90.

[2] Richardson, I. (2002). SPI Models: What Characteristics are Required for Small Software Development Companies?. *Software Quality Journal , 10* (2), 101-114.

[3] Kilpi, T. (1997). Product management challenge to software change process: preliminary results from three SMEs experiment. *Software Process: Improvement and Practice , 3* (3), 165-175.

[4] Nikula, U., Sajaniemi, J., & Kälviäinen, H. (2000). *A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises.* Lappeenranta, Finland: Telecom Business Research Center.

[5] Thörn, C., & Gustafsson, T. (2008). Uptake of modeling practices in SMES: initial results from an industrial survey. *International Conference on Software Engineering* (pp. 21-26). Leipzig, Germany: ACM New York, NY, USA.

[6] Vaishnavi, V., & Kuechler, W. (n.d.). *Design research in information systems.* Retrieved February 4, 2009, from Association for Information Systems: http://www.isworld.org/Researchdesign/drisISworld.htm

[7] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly , 28* (1), 75-106.

[8] Goel, A. K. (1997). Design, Analogy, and Creativity. *IEEE Expert: Intelligent Systems and Their Applications , 12* (3), 62-70.

[9] Weerd, I. v., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a Reference Framework for Software Product Management. *Requirements Engineering, IEEE International Conference on* , 319-322.

[10] Rautianen, K., Lassenius, C., Vähäniitty, J., Pyhäjärvi, M., & Vanhanen, J. (2002). A Tentative Framework for Managing Software Product Development in Small Companies. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02). 8*, p. 251. Washington, DC, USA: IEEE Computer Society.

[11] Nooteboom, B., Zwart, P., & Bijmolt, T. (1992). Transaction Costs and standardisation in professional services to small business. *Small Business Economics , 4* (2), 141-151.

[12] Fowler, P., Carleton, A., & Merrin, B. (1998). Transition Packages: An Experiment in Expediting the Introduction of Requirements Management. *Proceedings of the Third IEEE International Conference on* (pp. 138-145). Colorado Springs, Colorado: IEEE Computer Society.

[13] Kumar, K., & Welke, R. (1992). Methodology EngineeringR: a proposal for situation-specific methodology construction. In W. Cotterman, & J. Senn, *Challenges and strategies for research in systems development* (pp. 257-269). New York, NY, USA: John Wiley & Sons, Inc.

[14] Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology , 38* (4), 275-280.

[15] Harmsen, F., Brinkkemper, S., & Oei, J. L. (1994). Situational method engineering for informational system project approaches. In E. A. A. Verrijn-Stuart and T. W. Olle (Ed.), *Proceedings of the IFIP Wg8.1 Working Conference on Methods and Associated Tools For the information Systems Life Cycle (September 26 - 28, 1994). IFIP Transactions A-55*, (pp. 169-194). New York , USA: Elsevier Science.

[16] Harmsen, A. (1997). *Situational Method Engineering.* Moret Ernst & Young.

[17] Harmsen, F., Lubbers, I., & Wijers, G. (1995). Success–Driven Selection of Fragments for Situational Methods The S3 model. In K. Pohl, & P. Peters (Ed.), *Proceedings of the Second International Workshop on Requirements Engineering: Foundations of Software Quality*, (pp. 104-114). Aachen, Germany.

[18] Slooten, K., & Hodes, B. (1996). Characterizing IS Developments Projects. *Proceedings of the IFIP WG8.1 Conference on Method Engineering* (pp. 29-44). London, UK: Chapman and Hall.

[19] Bekkers, W., Weerd, I. v., Brinkkemper, S., & Mahieu, A. (2008). The Influence of Situational Factors in Software Product Management: An Empirical Study. *Proceedings of the 21th International Workshop on Product Management.* Barcelona, Spain.

[20] Engels, G., Lewerentz, C., Nagl, M., Schäfer, W., & Schürr, A. (1992). Building integrated software development environments. Part I: tool specification. *ACM Transactions on Software Engineering and Methodology (TOSEM) , 1* (2), 135-167.

[21] Visitacion, M., & Schwaber, C. (2005, Sep. 21). Integrate Requirements Management With SCM To Enable More Informed Project Management. *Forrester* , 3.

[22] Repenning, A., Ioannidou, A., Payton, M., Wenming, Y., & Roschelle, J. (2001). Using components for rapid distributed software development. *IEEE Software , 18* (2), 38-45.

[23] Nuseibeh, B., & Easterbrook, S. (2000). Requirements Engineering: A Roadmap. In A. Finkelstein (Ed.), *The Future of Software Engineering.* Special Volume published in conjunction with ICSE 2000.

[24] Crnkovic, I., Asklund, U., & Dahlqvist, A. P. (2003). *Implementing and Integrating Product Data Management and Software Configuration Management.* Norwood, MA, USA: Artech House, Inc.

[25] *ThoughtWorks Studios*. (n.d.). Retrieved April 28, 2009, from ThoughtWorks Studios: http://studios.thoughtworks.com/

[26] Britton, C., & Doake, J. (2000). *Object-Oriented System Development: A Gentle Introduction.* London: McGraw-Hill.

[27] Cannon, D., & Wheeldom, D. (2007). *Service Operation Itil* (Vol. Version 3). Stationery Office.

[28] Vähäniitty, J. (2006). Do small software companies need portfolio management, too?. *In Proceedings of the 13th International Product Development Management Conference* (pp. 1471-1486). Milan, Italy: EIASM.

[29] Cusumano, M. A. (2004). *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad.* New York, NY, USA: The Free Press.

[30] Brereton, P. (2004). The software customer/supplier relationship. *Communications of the ACM , 47* (2), 77-81.

[31] Wiegers, K. (1999). First things first: prioritizing requirements. *Software Development , 7* (9), 48-53.

[32] Natt och Dag, J., Gervasi, V., Brinkkemper, S., & Regnell, B. (2004). Speeding up Requirements Management in a Product Software Company: Linking Customer Wishes to Product Requirements through Linguistic Engineering. In J. Natt och Dag, V. Gervasi, S. Brinkkemper, & B. Regnell (Ed.), *Proceedings of the Requirements Engineering Conference, 12th IEEE International* (pp. 283-294). Lund Univ., Sweden: IEEE Computer Society.

[33] Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Natt och Dag, J. (2001). An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering* (p. 84). IEEE Computer Society.

[34] Akker, M., Brinkkemper, S., Diepen, G., & Versendaal, J. (2008). Software product release planning through optimization and what-if analysis. *Information and Software Technology , 50* (1-2), 101-111 .

[35] Martin, J. (1991). *Rapid Application Development.* New York: Macmillan.

[36] Agarwal, R., Prasad, J., Tanniru, M., & Lynch, J. (2000). Risks of rapid application development. *Communications of the ACM , 43* (11), 177-188.

[37] Beynon-Davies, P., & Holmes, S. (2002). Design breakdowns, scenarios and rapid application development. *Information and Software Technology , 44*, 579-592.

[38] Servoy Inc. (2008). *Servoy Cleint User's Guide.* Retrieved February 2, 2009, from http://www.servoy.com/docs/servoy_4/ServoyClient4UserGuide.pdf

[39] Niessink, F., & Vliet, H. v. (2000). Software maintenance from a service perspective. *Journal of Software Maintenance: Research and Practice , 12* (2), 103-120.

[40] Rosqvist, T., Koskela, M., & Harju, H. (2003). Software Quality Evaluation Based on Expert Judgement. *Software Quality Journal , 11* (1), 39-55.

[41] Kitchenham, B. A., Pickard, L., Linkman, S., & Jones, P. (2005). A framework for evaluating a software bidding model. *Information and Software Technology , 47* (11), 747-760.

[42] Hyrkäs, K., Appelqvist-Schmidlechner, K., & Oksa, L. (2003). Validating an instrument for clinical supervision using an expert panel. *International Journal of Nursing Studies 40 (2003) , 40* (6), 619-625.

[43] Sandelowski, M. (1999). The call to experts in qualitative research. *Research in Nursing & Health , 21* (5), 467-471.

[44] Beecham, S., Hall, T., Britton, C., Cottee, M., & Rainer, A. (2005). Using an expert panel to validate a requirements process improvement model. *Journal of Systems and Software , 76* (3), 251-275.

[45] Lynn, M. (1986). Determination and quantification of content validity. *Nursing Research , 6*, 382-385.

[46] Weerd, I. v., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, A. (2006). On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. *Proceedings of the International Workshop on Software Product Management* (pp. 3 - 12). Washington, DC, USA: IEEE Computer Society.

[47] Torchiano, M., & Morisio, M. (2004). Overlooked Aspects of COTS-Based Development. *IEEE Software , 21* (2), 88-93.

[48] Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems , 16*, 531-541.

[49] Natt och Dag, J., Gervasi, V., Brinkkemper, S., & Regnell, B. (2005). A Linguistic-Engineering Approach to Large-Scale Requirements Management. *IEEE Software , 22* (1), 32-39.

[50] Sommerville, I., & Sawyer, P. (1997). Viewpoints: principles, problems and a practical approach to requirements engineering. *Annals of Software Engineering , 3*, 101-130.

[51] Hoek van der, A., Hall, R. S., & Heimbigner, D. (1997). Software Release Management. *Proc. Sixth European Software Engineering Conference* (pp. 159-175). Berlin , Germany: Springer.

[52] Ruhe, G., & Saliu, M. (2005). The Art and Science of Software Release Planning. *IEEE Software , 22* (6), 47-53.

[53] Wiegers, K. (1999). Automating requirements management. *Software Development , 7* (7), S1-S6.

[54] *Atlassian*. (n.d.). Retrieved May 27, 2009, from Atlassian: http://www.atlassian.com/software/jira/

[55] *Active Reload, LLC*. (n.d.). Retrieved May 27, 2009, from Active Reload, LLC: http://lighthouseapp.com/

[56] Lindgren, M., Land, R., Norström, C., & Wall, A. (2008). Key Aspects of Software Release Planning in Industry. *Proceedings of the 19th Australian Conference on Software Engineering* (pp. 320-329). Washington, DC, USA: IEEE Computer Society.