Reducing Uncertainty in Software Projects: Proposal for A Product Envisioning Framework

Hanna Mäenpää
Department of Computer Science, University of Helsinki
Helsinki, Finland
hanna.maenpaa@cs.helsinki.fi

***Abstract*—This article describes a software product envisioning framework that has been created by studying four dominant product vision models from the disciplines of new product development and software engineering. Results from evaluating the framework in three cases are presented. The framework can be used to assist communicating high-level goals of a software development effort, observing decrease of uncertainty about the project's goals and hopefully also in finding an "agile optimum" for requirements engineering activities in software project front-end.**

I. Introduction

At the beginning of a new software project, uncertainty about the development effort's goals, scope and priorities is high []. Agile methodologies aim to produce a working software prototype as early as possible [1], and a software engineer should rapidly be able to envision not only the internal structure of the future system but also, comprehend constraints and goals that are likely to guide the development effort.

During the effort, software team and product owner may communicate with a large amount of different stakeholders such as project-, product- or program management, customer service staff, other software teams, subcontractors, user experience- and usability designers and also possibly also end-users themselves [2]. As agile principles emphasize close and frequent interaction with stakeholders [1], the ability to work collaboratively with people from varying backgrounds and professional disciplines has become valued.

Communicating opportunities and constraints created by architectural decisions and agreeing on development priorities require understanding of sagendas, aspirations and wishes of project stakeholders [3]. With an understanding of the interdisciplinary context that surrounds the development effort a product owner and his team may better distinguish dependabilities in between the stakeholders' goals [4] and help resolve conflicts [2].

Even though agile principles have long encouraged face to face communication and documenting only what is absolutely necessary [1], [5] they do not provide enough support for initial product planning. The need for tools to assist 'barely sufficient' software product design, such as the Product Vision, have began to be identified amongst agile software development practitioners [6]. The absence of tools to follow the changing product definition and project context have created knowledge- and communication gaps in between the fast-delivering agile development project and determined product planning [7], [8]. The misalignments have been reported to complicate predictive extra-project activities, such as scheduling, change- and resource planning [2].

II. Background

Nambisan and Wilemon (2000) surveyed the potentials for cross-domain knowledge sharing between Software Development (SD) and New Product Development (NPD) [**?**]. Even though the terminology used to describe the NPD and SD processes were very different, they found similar stages of idea generation, analysis, design, development, testing and implementation in the workflows of both schools of thought. Also conjunctive factors influencing the project trajectory and measures for success were found. The researchers concluded that as the foundation of both theories is strongly based on innovation management, the interdisciplinary gap can possibly be bridged by studying their complementarities [**?**].

Software Development (SD) is a special form of new product development (NPD) [10]. A new project in both disciplines starts with acquiring domain understanding, generating and validating ideas and outlining either the complete solution or first steps to build it. In NPD terminology, a project initiation is called "fuzzy front-end of innovation" [11] while in software engineering, the corresponding ambiguous, creative and non-determined pre-development period is called a "concept phase" [12].

A New Product Development process starts by identifying a promising ideas [9], choosing what exactly is to be built and create a definite enough specification needed to start development [9], [13].

A. Software Product Success

In project front-end, nonformity of ideas impedes proceeding to the development phase [9]. Success and performance of pre-development activities rely on the whole project teams ability to understand the sources and nature of uncertainty [9], [14]. Generating access to new knowledge and integrating it to the development process increases the chances of the project reaching it's goals [15]. However, knowledge created in the project front-end phase is fluid, unstructured and hard to measure [9]. In Table I, Wilemon and Kim (2002) specify the nature of qualitative transformation of knowledge that is

TABLE I
CHARACTERISTICS OF FRONT-END- AND DEVELOPMENT PHASES [9].

	Front End	**Development phase characteristic**
Idea quality	Propable, fuzzy	Determined, fixed, clear, specific
Quality of information for decision making	Qualitative, informal, approximate	Quantitative, formal, precise
Outcome (action)	Blueprint (diminished ambiguity for decision making)	Product (making it happen)
Width and depth of focus	Broad, thin	Narrow, detailed
Rejecting an idea	Easy	More difficult
Degree of formalization	Low	High
Personnel involvement	Individual / small team	Full development team
Budget	Small or none	Large designated
Management methods	Un-structural, experimental, creative	Structured, systematic
Damage if project is abandoned	Small	Substantial
Upper management commitment	None or small	Usually high

required for ending the project front-end planning and starting the development phase.

Low end-product performance have been identified as one of the consequence of performing the project front-end activities negligently [10]. Sufficient front-end evaluation contributes to the team being able to estimate the project scope, thus creating a basis informed project resource planning [9], [10], [16]. Table 2 explains common problems in project front-end activities and presents some of their outcomes on product performance.

Balancing between sufficient and overdone project front-end work is complicated. Common problems in software envisioning include starting the development effort without no vision of the outcome [3], [10]. At the other extreme, the development effort can be commenced by illustrating a prophecy vision, which is not in align with the software development team's ability to perform within a reasonable timeframe [3]. The product vision may be based on invalidated assumptions about the stakeholders or end-users, or solely on end-user aspirations, leading low stakeholder acceptance [10], [17], [18]. The front-end analysis may also be too thorough, leading to big up-front design, possibly stagnating the development effort [3], [11]

B. The Software Product Vision

It is hard to express a vague software idea. Describing a project goal often starts by describing software features, rather than outlining the underlaying reason why the software is being built. A product vision describes the high-level goal of the development effort - it's reason for existence [**?**], [3], [19], [20]. Making personal knowledge about the project's goals available for team lays the groundstone for achieving a common understanding [21]. Understandind the projects goals, constraints and priorities already early in the process contributes positively to the end-product success [9], [10].

Existing software product vision models provide concrete parameters that can be used to analyze the end-product [6], [11], [22]–[24]. Modeling the new software product development as a collection of problems to be solved increases the development effort's possibilities of avoiding them, thus helping the effort to first define, then achieve it's goals with more precision [15].

At the beginning of a software project, absence of ideas is rarely the problem: while stakeholders are eager to share their needs and aspirations, goal of the software project may become fuzzy. Restricting the ideas by setting a a clear goal for the development effort yields ideas that are of better quality [17]. A well-defined and concise vision statement also helps the development team to distinguish the most critical building blocks of the software product [25].

The problem of choosing the most valuable features withholds and designing a future roadmap for the software product bare the most important strategic decisions in new product development. A mutual understanding of goals helps the product owner and his development team in aligning technological decisions to stakeholder goals, such as business requirements. It also facilitates the discussion and alignment of intra-project activities to project long-term goals and allows to prepare tactics for reacting to possible disruptions emerging from e.g. new technologies or changes in the marketplace [4].

Despite it's widely recognized benefits, the concept of a product vision has traditionally been present only at levels of company strategy- and product portfolio management [26]. It has been recognized only recently that agile development methods should be extended by tools for long term product planning [27] and methods for tracking the high-level product concept and it's underlying goals have started to emerge [3], [6], [26].

III. THE SOFTWARE PRODUCT ENVISIONING FRAMEWORK

In this chapter, a new framework for software product envisioning is introduced. The framework has been built as a synthesis of a four major, structured product vision statements from the disciplines of New Product Development and Marketing. The product vision models that lay the foundation of this work are:

- Cooper et al. Integrated Product Description [11]
- Geoffrey Moore's Product Vision Statement [22]
- Alex Osterweilder's Business Model Canvas [23]

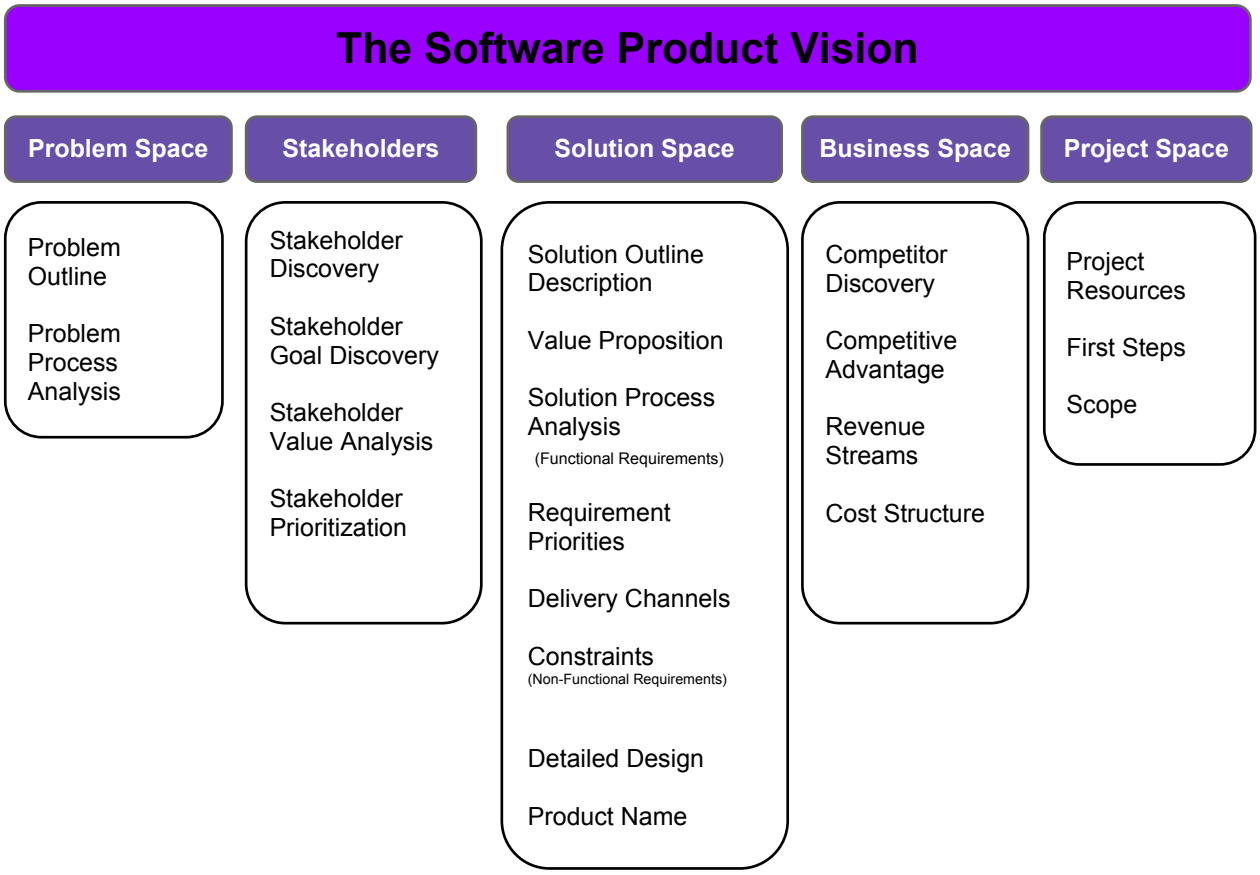


Fig. 1. The Software Product Envisioning Framework

TABLE II
REASONS FOR LOW PRODUCT PERFORMANCE [LSD]

Problem	Outcome
1. Missing of a compelling value proposition	Lack of customer interest
2. Neglect of defining a competitive advantage	Poor differentiation
3. Insufficient innovation front-end work	Slowed development due vision inclarity
4. Insufficient voice-of-customer evaluation	Poor user acceptance
5. Lack of end-user validation of concept	Poor user acceptance
6. Changes in product specifications	Delays due to misunderstandings
7. Variance in scope	Difficulty of estimating amonut of work
8. Lack of cross-functionality in team	Decreased team cohesiveness
9. Under-resourcing the team	Delays due task switching
10. Lack of competencies	Inability to proceed with the project

- Ash Mayura's Lean Canvas [24]

Various short, i.e. one-sentence descriptions describing a high-level product vision were found in agile software development literature [28]–[31]. However, because they provide very little contribution on project front-end analysis, they were not included in this comparison. While the Kano-model of product development [32] is a tool for evaluating product construct and steering the development effort, it was left out of this comparison for it's narrow focus on customer/end-user driven value prioritization of product features and attributes. Holtzblatt's "Contextual Design" -method [33] was also rejected from this study because of it's heavy emphasis on detailed software design.

To reflect on integrating the product envisioning process with the Agile Software Development process, a novel and unpublished envisioning framework by Agile Product Ownership author and industry expert Roman Pichler was also included in the comparison [6].

Each of the product envisioning models has their own emphasis. Cooper's Integrated Product Definition (IPD) bares only very general product vision parameters. While capturing a

high-level view on the end-product, it does not directly address analysis of the problem domain. When applying the heuristics of Osterweilder's Business Model Canvas (BMC) into software product discovery, a strong emphasis is paid on identifying customers, their needs, solution processes, resources and the business model. Lean Canvas has more emphasis on defining a strong competitive advantage than BMC and is also described to define the project scope: a "pathway to Minimal Viable Product". Geoffrey Moore's Product Vision Statement has been commonly used in agile software development literature. It captures the product vision at lesser tangible terms than the BMC and LC. Similarly to Cooper's Integrated Product Vision, Moore's vision statement does not encourage continuing problem- or solution domain analysis to process level, thus possibly contributing to starting the development effort based on assumptions.

Pichler's two-step method aims for integrating product envisioning into the agile software development process. It's first part, the Product Vision Board (PVB), encourages discovering end-users, their needs and the value that the software solution will provide to them. The second part, Product Canvas (P.PC) is a tool for transforming the high-level product vision, into a concrete development plan. It utilizes the Agile Modeling -method's [**?**] stepwise refinement of user stories. The model leaves more detailed problem/solution analysis and design to later stages of the development effort. Similarly, detailed project space analysis: scope, resources and first steps on starting the development effort are out of scope of the two tools.

Even though most of the already established vision statements were described with different vocabulary, they evoke coterminous knowledge in the product envisioning process. Therefore, each model and it's related instructions were studied to find similarities and differences. The analysis yielded 18 underlying parameters were chosen and organized into five integrative meta-categories (Problem Space, Stakeholders, Solution Space, Market Space and Project Space). One parameter, Detailed Design, was added to identify whether the teams were starting to be close to entering the development phase. Figure 1 demonstrates resulting framework.

IV. Action Research Case Study

The validity of the new Software Product Envisioning Framework was been evaluated by performing action research three cases. The author arranged a software innovation workshop and two project courses for computer science major students to test validity of the software product envisioning framework. She participated in all cases the course in the role of a facilitator.

A. Software Innovation Workshop

Software Innovation Workshop was a 17-day course that was arranged at University Of Helsinki's Department of Computer Science in May 12. Action research aimed to identify, which product envisioning framework parameters would be discovered in a software product design process that started maximum uncertainty about both the goal and means to achieve it. Analysis of some parameters was evoked during the workshop by setting clear goals for each session, some were left to be discovered by the students themselves.

During the workshop, teams of 3-4 students worked collaboratively to create novel software ideas. The curriculum was designed to conform to three stages: discovery, scoping and building a business case [11]. Schedule of the five workshops was designed to be intensive, consisting of five, mandatory 4-hour workshops. In between the workshops, the teams spent 2-4 days contemplating the challenge, interviewing people, validating their assumptions and studying the phenomena related to their teamwork.

Outline of the workshop (WS) topics and homework (HW) assignments:

- HW.1 Discover new, exciting technologies.
- WS.1 Perform a Persona interview [**?**]. Find a problem to solve. Identify stakeholders and their needs. Outline a solution.
- HW.2 Study competing solutions to the same problem.
- WS.2 Iterate your solution, find a means to differentiate it from competitors.
- WS.3 Analyze delivery channels and benefits for stakeholders. Define a competitive advantage.
- HW.3 Find a critical feature of the system [25].
- WS.4 Identify possible partners in providing the service. Identify revenue streams. Create a product backlog.
- WS.5 Concept demo

Goals of each workshop were introduced at the beginning of sessions. Participants were encouraged to facilitate their communication by using both software modeling tools and free form sketches. Clean guidelines were intended to be provided for accomplishing the tasks, but during the first session it became obvious that the students did not need detailed guidance. At third workshop, students were confused about where the modeling actions would lead. Therefore a product vision canvas (Figure 2) was introduced. Table 3 displays the parameters that were discovered during the software product design process.

When asked, some Innovation Workshop participants implied that they had realized the wider context a software can have. After the initial confusion, seeing the canvas had helped them to concretize a goal for the software discovery effort.

B. Open Data -Application Development

To evaluate relevance of the framework in a real software project with highly experienced developers and moderate uncertainty, a startup -style software product development effort was started as a masters' level project course. Four professional web- and mobile application developers were assigned to create an open data application for Apps4Finland [**?**]-competition. The team consisted of an User Experience Designer (UXD) and three programmers.

After an free form ideation session, the UXD presented an initial concept of a mobile, map-based rest stop application as concept sketch, containing images and user interface mockups. Researcher observed which product vision parameters were intuitively included in the presentation. Development effort started quickly and was mostly guided by online discussion and light sketches of user interfaces. No detailed design modeling was made either in the project front-end, during development or post-project. A summary of Software Product Vision Parameters that were discovered the three project stages are found in table IV. Post-project parameters were qualitatively surveyed from the participants, yielding very unanimous answers.

Of the Product Envisioning Framework parameters Cost Structure, Revenue streams and non-functional requirements were not addressed at all. Stakeholder value analysis and defining a strong competitive advantage were addressed very lightly.

Even though the application was professionally developed and launched in in social media and received wide public attention in the Apps4Finland competition, it did not perform well. When analyzing low product performance with the user experience designer after the project had been ended he identified that the product was terminated too early. He expressed that continuing development, planning determined marketing activities and launching the product with better timing and visibility could still bring the product to its target audience successfully. It is impossible to say, weather more thorough analysis of the competitive environment would have contributed to the low success of the end-product.

C. Robotics Lab

The Robotics Lab -project was started for developing an embedded system as a showcase for robotics programming education. Participants of the project included three Computer Science major students and one Physics student who all shared extensive knowledge in DIY-robotics and programming. One of the participants was assigned as a project manager.

The core project team spent autumn 2012 investigating hard- and software technologies related to theit topic, building prototypes of various subsystem options, including speech recognition, movement on demand and capacitative sensing. Processes of manufacturing custom circuitboards and chassis parts were also studied.

During the extensive project front-end activities, the team gained advanced knowledge about the solution domain and many small sub-project ideas started to emerge. While ”big picture” of the projects goals was clear, plans for actualizing the steps for proceeding become vague. To help the project team align their actions according to original goals and to prioritize the small team’s efforts. Eight parameters of the framework were chosen and composed into a simple A3 canvas tool (Figure 2). Parameters that were chosen to be used in the canvas were:

- Problem Outline
- Stakeholder Discovery
- Solution Outline
- Competitive Advantage
- Solution Process Analysis
- Requirement Priorities
- Next Steps

When sketching the sub-projects, discussion was mainly dominated by solution space details. However, when asked, the team was able to deliver information related to all parameters.

After the meeting, the subproject sketches were reviewed with the project group. Three major aggregate categories were identified: Platform Projects, Vision/Hearing Interfaces and Small Application Projects. Each category contained 3-4 project sketches. Platform Projects -category was chosen to be top priority. After the group meeting, the project manager was invited separately to review the top priority sketches and was interviewed about the experience of using the framework in capturing the project ideas. He indicated that goals of each

TABLE III
CASE A: SOFTWARE PRODUCT ENVISIONING FRAMEWORK PARAMETERS DISCOVERED AT EACH PHASE OF THE INNOVATION WORKSHOP.

	Parameter
Pre-workshop task (HW0)	-
Workshop 1	Problem outline, Problem Process Analysis, Stakeholder Discovery, Stakeholder Goal Discovery, Solution Outline
HW1	Competitor Discovery
Workshop 2	Value Proposition, Stakeholder Value Analysis, Competitive Advantage, Solution Process Analysis
Workshop 3	Delivery Channels, Stakeholder Prioritization, Requirement Priorities
HW3	First Steps
Workshop 4	Project Resources (partnering), Revenue Streams
Workshop 5	-

TABLE IV
CASE B: SOFTWARE PRODUCT ENVISIONING FRAMEWORK PARAMETERS DISCOVERED IN OPEN DATA APPLICATION DEVELOPMENT PROJECT.

	Parameter
Project Front-end	Problem Outline, Stakeholder Discovery, Solution Outline, Competitor Discovery, Delivery Channels, Product Name, Solution Process Analysis, First Steps
During Development	Problem Process Analysis, Stakeholder Goal Discovery, Functional Requirements, Value Proposition, Stakeholder Prioritization, Project Resources, Project Scope
Post-development interview	Product name, Problem Outline, Solution Outline, Stakeholder Discovery, Value Proposition, Competitive Advantage, Prioritization Criteria

subproject had become clearer and he had gained an overview of priorities for next steps of the project.

V. Discussion

In Case A Software Product Envisioning Framework was used to observe the decrease of uncertainty in new fuzzy front-end of software product innovation. In Case B, parameters were used in observing the amount of uncertainty about a real software development project's goals before, during and after the development effort. In third case, chosen parameters of the framework were used to accelerate starting the development phase after a very long fuzzy front-end of innovation.

Cases A and C imply that visualizing parameters on the form of a canvas poster clarifies the output of the development effort. By requiring a minimum amount of parameters to be present in software project front-end, starting the development effort with insufficient vision can be avoided. On the other hand, restricting the amount of parameters helps avoiding a big up-front design.

Project B was market-driven. Project C was driven by technology opportunities. Therefore, applying some of the parameters (revenue streams, competitive advantage) was not applicable. In Case A, the Value Proposition was used unanimously with the Product Name. As a conclusion, fixing a predetermined set of parameters into a canvas tool and using it for all purposes may restrict it's compatibility for different kinds of projects and is therefore discouraged by the author.

Requirement prioritization activities in case A yielded end-user value driven prioritization. In case C, the same parameter evoked thorough analysis on sub-project dependencies. This implies that choosing a general enough parameter name increased it's compatibility for different kinds of projects.

Some of the Software Product Envisioning Framework parameters were present from the very beginning of all case studies. From all parameters, constraints was the one that was least addressed. This implies that elicitating constraints, i.e. "Non-Functional requirements" may require more thorough guidance.

Three of the established product envisioning models contain a design heuristic that can be used to aid design. In all cases the new framework was evaluated in, the design process was highly iterative. A predefined heuristic worked well for guiding the product discovery novices at Innovation Workshop course in Case A. Remarkably with experienced developers performing only a short project front-end before starting development (case B), the design process could not be directed by suggesting the team to discover parameters. Therefore, the framework was used to follow the product discovery process i.e., the decreasing uncertainty about the project's goals.

Some of the concepts introduced to the computer science major students were new, originating from the world of marketing and product development. However, the concepts of "value proposition", "competitive advantage" and "revenue streams" were easily digested after a brief explanation. Using the framework in the context in case A was thus, Computer Science education applicable.

VI. Conclusions

Uncertainty about a software project's goals and priorities complicates development efforts. Uncertainty is often caused by the volatility of the software project's environment: varying stakeholder goals or changes in the market- or technology space. Intra-project factors such as project management or competences of the product owner and the software team also effect the development effort. Uncertainty can be reduced by creating new knowledge [21].

While the emphasis of agile methods are focused on a single development team working on a single project [27] and close collaboration of the development team and it's customer [1], they currently lack support for initial and long-term product envisioning [2]. Predictive product management is complicated by the evolving release scope that is typical for agile development methodologies. Knowledge gaps in between long-term planning, daily development activities and user experience design is reported to distract determined product development [2], [7], [8].

Researchers and practitioners have awaken to recognize product envisioning and -planning as an amendment to the agile development process. Some product envisioning frameworks exist in New Product Development body of knowledge [11], [22]–[24]. An Agile Product Management practitioner expert has published his initial, two-step agile product envisioning process [6].

When analyzing the existing product envisioning methods, significant similarities were found. All methods approach a great deal of the same questions with different terminology. Therefore, as a contribution of this thesis, an interdisciplinary framework for software product envisioning terminology is created by comparing the underlaying knowledge each product envisioning method evokes.

The framework was evaluated in a university course of innovative product discovery for software engineers, as well as in two capstone projects. The results from the experiments imply that the software engineering curriculum can be complemented with pre-project activities assisted by the product envisioning framework. In case B, the framework was applicable for evaluating the amount of uncertainty about the project's goals was present at a software project. In case C, the framework was successfully used to capture vague ideas and thus reduce uncertainty about a project's goals.

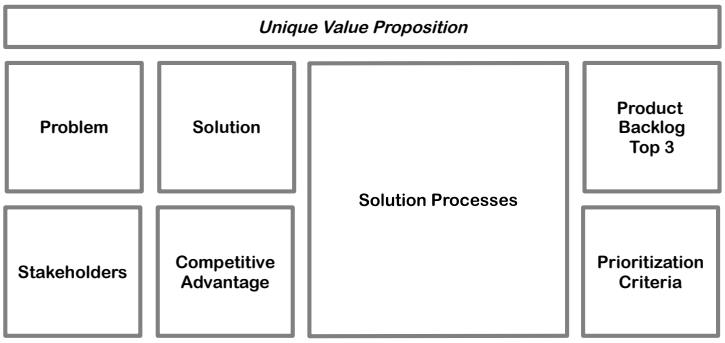
While uncertainty in software projects is a chaotic and complex phenomenon, no "silver bullet" can address all situations. Therefore, the framework parameters are recommended to be chosen to suit the project at hand. Parameters can be accompanied with a simple, one-page template to help a software development team to find an "agile optimum" for requirements engineering activities in the project front-end.

References

[1] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries,

TABLE V
THE CANVAS TOOLS USED IN CASES A AND C

3. Value proposition	
1. Problem	2. Stakeholders
4. Solution	5. Competitive advantage
6. Solution processes	
7. Revenue streams	8. Product Backlog top 3



J. Kern, B. Marick, R. C. Martin, S. Mallor, K. Shwaber, and J. Sutherland, "The Agile Manifesto and The Twelve Principles of Agile Software," The Agile Alliance, Tech. Rep., 2001. [Online]. Available: http://www.agilemanifesto.org/

[2] D. Karlstrom and P. Runeson, "Combining agile methods with stage-gate project management," *Software, IEEE*, vol. 22, no. 3, pp. 43–49, 2005.

[3] R. Pichler, *Agile Product Management with Scrum: Creating Products that Customers Love*. Pearson Education, 2010. [Online]. Available: http://books.google.co.uk/books?id=aLSu0P0EojEC

[4] B. Boehm, "Making a difference in the software century," *Computer*, vol. 41, no. 3, pp. 32–38, 2008.

[5] S. Ambler, *Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process*, ser. Programming, software development. Wiley, 2002.

[6] R. Pichler, "All things product owner," 2013. [Online]. Available: http://www.romanpichler.com/blog/

[7] N. Dzamashvili Fogelstrom, T. Gorschek, M. Svahnberg, and P. Olsson, "The impact of agile principles on market-driven software product development," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 22, no. 1, pp. 53–80, 2010.

[8] C. Manteli, I. van de Weerd, and S. Brinkkemper, "Bridging the gap between software product management and software project management," in *Proceedings of the 11th International Conference on Product Focused Software*. ACM, 2010, pp. 32–34.

[9] J. Kim and D. Wilemon, "Focusing the fuzzy front–end in new product development," *R&D Management*, vol. 32, no. 4, pp. 269–279, 2002.

[10] M. Poppendieck and T. Poppendieck, *Implementing Lean Software Development: From Concept to Cash*, ser. The Addison-Wesley Signature Series. Addison-Wesley, 2007.

[11] R. Cooper and S. Edgett, *Lean, Rapid and Profitable New Product Development*. CreateSpace, 2009.

[12] "Ieee standard glossary of software engineering terminology," *IEEE Std 610.12-1990*, p. 1, 1990.

[13] F. P. Brooks, "No silver bullet: Essence and accidents of software engineering," *IEEE computer*, vol. 20, no. 4, pp. 10–19, 1987.

[14] A. Pearson, "Managing innovation: an uncertainty reduction process," *Managing innovation*, pp. 18–27, 1991.

[15] W. A. Sheremata, "Finding and solving problems in software new product development," *Journal of Product Innovation Management*, vol. 19, no. 2, pp. 144–158, 2002. [Online]. Available: http://dx.doi.org/10.1111/1540-5885.1920144

[16] IEEE Computer Society, *Software Engineering Body of Knowledge (SWEBOK)*, P. Bourque and R. Dupuis, Eds. EUA: Angela Burgess, 2004. [Online]. Available: http://www.swebok.org/

[17] M. Pikkarainen, W. Codenie, N. Boucart, and J. A. Heredia, *The Art of Software Innovation - Eight Practice Areas to Inspire your Business*. Springer, 2011.

[18] C. Kessler and J. Sweitzer, *Outside-In Software Development: A Practical Approach to Building Successful Stakeholder-Based Products*, ser. Networking Technology. IBM Press/Pearson, 2008.

[19] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2009.

[20] P. Trott, *Innovation Management and New Product Development*, ser. Pearson education. Financial Times Prentice Hall, 2008. [Online]. Available: http://books.google.co.uk/books?id=9hv4GqUq1E0C

[21] I. Nonaka and H. Takeuchi, *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, USA, 1995.

[22] G. A. Moore, *Crossing the chasm*. New York: Harper Business, 1991.

[23] A. Osterwalder and Y. Pigneur, *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley, 2013.

[24] A. Maurya, *Running Lean: Iterate from Plan A to a Plan That Works*, ser. Lean Series. O'Reilly Media, Incorporated, 2012. [Online]. Available: http://books.google.co.uk/books?id=j4hXPn233UYC

[25] B. Boehm, "Value-based software engineering: reinventing," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 2, p. 3, 2003.

[26] J. Vähäniitty, "Towards agile product and portfolio management," 2012.

[27] P. Kettunen and M. Laanti, "Combining agile software projects and large-scale organizational agility," *Software Process: Improvement and Practice*, vol. 13, no. 2, pp. 183–193, 2008.

[28] L. Hohmann, *Innovation Games: Creating Breakthrough Products Through Collaborative Play*. Addison-Wesley, 2007.

[29] J. Highsmith, *Agile Project Management: Creating Innovative Products*, ser. Agile Software Development Series. Pearson Education, 2009. [Online]. Available: http://books.google.co.uk/books?id=VuFpkztwPaUC

[30] *Agile Estimating And Planning*. Pearson Education, 2006.

[31] D. Leffingwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Pearson Education, 2010.

[32] N. Kano, "Upsizing the organization by attractive quality creation," in *Total Quality Management Proceedings of the First World Conference*. Carfax Publishing UK, 1995, pp. 60–72.

[33] K. Holtzblatt, J. Wendell, and S. Wood, *Rapid Contextual Design:*, ser. Morgan Kaufmann Series in Interactive Technologies. Elsevier Science, 2005. [Online]. Available: http://books.google.co.uk/books?id=VjO6n9stHzUC