
Hanna Mäenpää

Helsinki April 16, 2013
MSc Thesis
UNIVERSITY OF HELSINKI
Department of Computer Science
The change from pre-descriptive "waterfall" software process into iterative and incremental models has created a need for redefinition of software requirements engineering. Agile methodologies have emerged to support the paradigm shift by treating the symptoms: emphasizing change management and customer collaboration to embrace volatility of requirements and priorities whilst in development. However, it has been recognized that fast-paced agile development does not provide sufficient support for initial or long-term planning of the software product. Research and practitioner literature have started to address the need with the concept of a high-level definition of the software project’s outcome: the software Product Vision.

In this thesis, uncertainty in new product development is studied from the perspective of Innovation Management. As a vehicle for reducing uncertainty in software projects, the concept of an software Product Vision (reason for the project’s existence) is chosen to be examined from the viewpoints of New Product Development and Software Engineering literature. The work describes sources of uncertainty in software projects and explains the effects of a mutually understood software Product Vision on software project performance and end-product acceptance. Key parameters for an interdisciplinary and unified software Product Vision are identified by studying four existing and one emergent Product Vision models. Finally, a new Product Vision framework (InnCa) is created based on semantic analysis. The framework’s applicability on software projects is evaluated in three participatory action research -case studies.

As a result, it is concluded that common parameters of a interdisciplinary "Product Vision" can be identified. The framework created can be used to ideate, rapidly capture, iterate and analyze vague software ideas. It is applicable for sharing knowledge about the project’s high-level goals amongst the project’s stakeholders.

However, it is not argued in this thesis that the framework could be used in all kinds of projects and circumstances. While uncertainty in software projects is a chaotic and complex phenomenon, no "silver bullet" can address all situations. The topic of software Product Vision may prove grounds for further research, possibly leading to practical tools for assessing and quantifying uncertainty about goals during a software project’s trajectory.
## Contents

1 **Introduction**  
  1.1 Research Questions ........................................ 4  
  1.2 Scope ..................................................... 4  
  1.3 Thesis Structure ........................................... 5  

2 **Uncertainty in Software Projects**  
  2.1 Understanding Uncertainty .................................... 8  
  2.2 Uncertainty in Software Projects ............................ 12  
  2.3 Software Project Success .................................... 14  
    2.3.1 Project Front-End ..................................... 17  
    2.3.2 Initial Product Planning ................................. 19  
    2.3.3 Long Term Product Planning .............................. 20  
  2.4 The Software Product Vision .................................. 23  
    2.4.1 Product Envisioning Benefits ............................ 24  
    2.4.2 Product Envisioning Problems ............................ 25  
    2.4.3 Established Product Vision Models ....................... 27  
    2.4.4 Experimental Product Vision Models ...................... 35  
  2.5 Comparison of the Product Vision Models ...................... 39  
  2.6 Conclusion ................................................ 42  
  2.7 Summary .................................................. 46  

3 **Constructing the new Software Product Vision -framework (InnCa)**  
  3.1 Product Vision Canvas (InnoCanvas) ......................... 52  

4 **Evaluating the Framework**  
  4.1 Case Selection Strategy ...................................... 53  
  4.2 Case A:  
    Software Innovation Workshop ................................ 54
4.3 Case B:  
Open data -application "Taukkis" .................................... 60

4.4 Case C:  
Robotics Lab .......................................................... 66

5 Discussion 70

5.1 Analysis ............................................................. 70

5.2 Implications ......................................................... 72
  5.2.1 Uncertainty About Project Goals in Cases .................... 73
  5.2.2 Adjusting Parameters of the InnoCanvas Tool ............. 74
  5.2.3 Amount of Active Product Vision Parameters in Product Envisioning ................................................. 75
  5.2.4 Value of Generalization in InnCa-Framework’s Parameters .. 76

5.3 Limitations of results .............................................. 77

5.4 Future Work ......................................................... 78
  5.4.1 Shared mental models about software project’s goals. .... 78
  5.4.2 Modern Software Requirements Engineering as a complex system of collaborative learning .................... 80

6 Conclusions 81

References 83

Appendices

A Mapping of terms in between the proposed and existing Product Vision statements

B Aiding questions for Applying the Software Product Vision Framework Parameters

C Process description from the Innovation Workshop (Case A)

D Two example product concepts from the Innovation Workshop
(Case A)

E Two example canvases from Case C

F Reduction of uncertainty in case A (Software Innovation Workshop) as a subjective evaluation by the Author.

G Perceived uncertainty by the software team in case B (Open Data Application Development) before and after the development phase.

H Perceived uncertainty by the Project Manager in case C (the Robotics Lab) before and after the facilitation

I Questionnaires for evaluating perceived uncertainty in Case B
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer, customer representative</td>
<td>A person or an organization which benefits directly from building the new product or software.</td>
</tr>
<tr>
<td>End-user, user</td>
<td>A person who is using the product, service or software.</td>
</tr>
<tr>
<td>An Innovation (noun)</td>
<td>A newly introduced, improved product, service, method or process.</td>
</tr>
<tr>
<td>To innovate (verb)</td>
<td>To strive for creating or introducing something innovative, to initiate, pioneer, debut.</td>
</tr>
<tr>
<td>New product development (NPD)</td>
<td>The act of developing a new product, service or process that is well defined, implemented and utilized, either commercially or for the common good.</td>
</tr>
<tr>
<td>Product</td>
<td>Outcome of new product development efforts. A product may consist of a combination of physical, service, process or software elements.</td>
</tr>
<tr>
<td>Product Backlog</td>
<td>A list of items that is prioritized according to goals of the software development effort.</td>
</tr>
<tr>
<td>Product Discovery</td>
<td>Creating a high-level vision for a software product.</td>
</tr>
<tr>
<td>Product Management</td>
<td>The actions needed for guiding development work according to the software product stakeholders’ goals.</td>
</tr>
<tr>
<td>Product Owner</td>
<td>A person who represents software customer and other stakeholder needs and expectations to the development project. The product owner is held in account for the success of a project.</td>
</tr>
<tr>
<td>Product Vision, High-Level Vision</td>
<td>A description about a software project’s goals.</td>
</tr>
<tr>
<td>Software product</td>
<td>Outcome of the software project. May be e.g. a commercial product, software component or a prototype.</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Any party that has requirements for the outcome of the project. A stakeholder may be a person or system that has a relation to the product, service or a system to be built. The relation may be e.g. interest or dependence.</td>
</tr>
<tr>
<td>Stakeholder value</td>
<td>Benefit of using the system for a stakeholder.</td>
</tr>
</tbody>
</table>
1 Introduction

The ongoing transition from traditional plan-based software development to iterative and incremental process models has created a need for redefinition of software Requirements Engineering (RE). While Agile software processes aim to produce a working prototype as early as possible, a software developer is expected to be able to very rapidly identify and capture requirements for the new software to be built. Together with the Product Owner, he should evaluate and communicate opportunities and limitations given by technological range and actualize stakeholder goals and constraints into a baseline architecture that fulfills its purpose sustainably. In this task, the developer benefits from understanding the multidisciplinary context that surrounds the development effort.

As agile principles emphasize frequent face-to-face communication and fast-paced stakeholder collaboration above written documentation, stakeholders may lose the "big picture" of the software project’s goals during the upbeat development effort. While current project planning tools such as the product backlog do not support whole-product thinking, gaps in feedback loops in between daily software development, user experience design and long term product planning have been reported extensively in research literature.

A 'Product Vision' outlines the objective of a development effort - the reason for a software project’s existence. In New Product Development, achieving a shared mental model of a project’s goals, its constraints and priorities early in the process has been identified as a key factor for end-product success. The concept of a mutually understood Product Vision has traditionally been present only at the levels of portfolio- and product planning. However, it is argued in this thesis that the concept of a software Product Vision can be extended to the levels of software projects and related short- and long-term planning.

The Product Vision can be stated in terms of the whole software product and complemented by envisioning the goals of each distinct sub-project that are required to build it. A shared understanding of the interdisciplinary context that surrounds the development effort allows the software development team to enable informed negotiation development priorities and software design decisions. It helps the development
team to create the "right product" by ensuring alignment of intra-project activities with the most important stakeholder goals, also allowing the team to prepare better for disruptions that e.g. emergent technologies or changes in the marketplace may create during the development effort.

In this thesis, uncertainty in new software product development is studied from the perspective of Innovation Management. As a vehicle for reducing uncertainty about project goals, the concept of an software Product Vision (reason for the project’s existence) is chosen to be examined from the viewpoints of New Product Development and Software Engineering literature. Key parameters for an interdisciplinary software Product Vision are identified by studying four existing and one emergent Product Vision models. Finally, a new Product Vision framework (InnCa) is created based on semantic analysis. The framework’s effect on software projects in evaluated in three participatory action research -case studies.

As a result, it is concluded that the framework created can be used to ideate, rapidly capture, iterate and analyze vague software ideas. It is applicable for sharing knowledge about the project’s high-level goals amongst the project’s stakeholders. Most importantly it is designed to gently guide collaborative creativity into a productive direction, resulting in a result that is compatible with starting an agile software development effort.

It is not argued in this thesis that the framework could be used in all kinds of projects and circumstances. While uncertainty in software projects is a chaotic and complex phenomenon, no "silver bullet" can address all situations. Instead, creating an ontology around the topic of the software Product Vision can prove grounds for further research, helping to assess and quantify of uncertainty during the software project’s trajectory. Meanwhile, applying the framework is hoped to help to small, fast-paced agile teams and their stakeholders to capture vague software ideas and to understand the software project’s near-future goals.
1.1 Research Questions

The need for front-loading the software project with a thorough analysis and emphasizing holistic product thinking have been extensively reported in literature. However, no unified concept or methodology for doing so exists. For it’s widely recognized value in strategic management, new product development and product portfolio management, the concept of "Product Vision" is taken into observation from the viewpoint of a single software project. The research questions are thus set as:

RQ1: What is known about the concept of Product Vision in NPD and SE literature?

RQ2: How to construct a holistic software Product Vision framework?

RQ3: How can the constructed software Product Vision framework be applied in software development efforts?

RQ1 and RQ2 were addressed by reviews on classic 1) Innovation Management, 2) New Product Development and 3) Agile Software Development literature. A literature review on Product Vision was also conducted in Software Engineering scientific body of knowledge. Keywords for the search were:

("Product Vision" AND agile AND "requirements engineering" AND software) OR
("Product Vision" AND agile AND "uncertainty")

RQ3 was studied by conducting action research in three cases, where the software Product Vision framework was applied and adjusted in different settings. Case selection strategy is further elaborated in section 4.1.

1.2 Scope

Even though knowledge, strategic- and product portfolio management provide essential input to new product development, analysis of their processes is out of scope for this thesis. Author acknowledges that meta-models, prioritization schemes and even algorithms to assist backlog grooming have been developed but also, their analysis is out of the scope of this thesis.
Instead, this work focuses on enhancing communication of small software teams with it’s stakeholders. Special emphasis is paid on accelerating the software innovation process by creating new knowledge from the extra-project perspective to to the software team to help the create a shared mental model in between the stakeholders about the overarching goal of the development effort - a software product.

1.3 Thesis Structure

Chapter 2 reviews the sources and effect of uncertainty in software projects, based on Innovation Management, New Product Development and Agile Software Development literature. In Chapter 2.3, existing Product Vision templates are introduced.

Chapter 3 analyzes similarities and differences of Product Vision frameworks. Chapter (3.2) introduces a newly constructed Product Vision framework (InnCa).
Chapter 4 presents participatory action research case studies aiming for evaluation of the framework in software projects.
Chapter 5 discusses the results. Chapter 6 summarizes conclusions.

Figure 1 explains the relationship between chapters.
Figure 1, Structure of the thesis and relations of chapters.
2 Uncertainty in Software Projects

Nambisan and Wilemon (2000) surveyed the potentials for cross-domain knowledge sharing between Software Development (SD) and New Product Development (NPD) [NW00]. 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 [NW00].

In both New Product Development (NPD) and Software Engineering (SE) a project starts with acquiring domain understanding, generating and validating ideas and outlining either the complete solution or first steps to build it. In NPD terminology, the project initiation is called "Fuzzy front-end of innovation" [Tro08] while the corresponding ambiguous, creative and non-determined pre-development phase in Software Engineering is called "requirements elicitation and analysis" [Som07].

In this chapter, sources of uncertainty are examined from the viewpoint of innovation management (2.1), New Product Development and Agile Software Development disciplines (2.2). Effects of uncertainty are then reflected on software project performance and end-product acceptance and finally the significance of a collaboratively created and understood software Product Vision is highlighted (2.3). Benefits of Product Vision on software product and project performance are then examined (2.4.1). Existing Product Vision models are examined from the viewpoints of New Product Development (2.4.3) and Software Engineering (2.4.4) literature.

Chapter 2.5 summarizes the conclusions.
2.1 Understanding Uncertainty

Innovation management provides a theoretical ground for understanding sources and effects of uncertainty that affect software project’s trajectory and end-product success. Bareghheh et al. [BRS09] studied remarkable definitions of innovation in literature over the time from 1934 to 2008. The researchers chose quotations from experts of economy, entrepreneurship, innovation, marketing and organizational study. They analyzed the worlds of e.g. Porter, Schumpeter, Drucker and 60 other grand names in order to find an interdisciplinary definition for the word. Content analysis revealed the wording:

‘Innovation is a multi-stage process whereby organizations transform ideas into new/improved products, service or processes, in order to advance, compete and differentiate themselves successfully in their marketplace’ [BRS09].

Having to go through such a great endeavor to define innovation reveals the multi-lateral nature of the concept. The art of innovation requires a unique combination of theory, technology and business [Tro08], s.24. Depending on the viewpoint, innovativeness can be judged by novelty, technical possibilities, market reach, differentiation from competitors or overall impact on the economy.

The Innovation Ecosystem

Theorists have long strived for explaining the driving force for successful innovations. Early models developed in 1950-1970 described new products to be born from discovery of new technological advances, pushing new products to markets. Later, as consumers began to demand for differentiated products, a new theory of market pull was discovered [Zmu84]

In 1985, Rothwell and Zegweld described the innovation phenomenon as a highly networked process in between the innovator, society, science, technology and the marketplace [RZ85]. Figure 2 describes the interactive model of innovation as relationships between organizational capabilities and their surroundings. The model describes a concurrent, bi-directional learning-process: ideas and commercial products are described to being effected by- and influential to developments in science,
technology, society and the marketplace. Neither is the reality in software projects safely outlined by the goal of creating "just technological solutions". A wider understanding of the problem domain is required for also, a successful software design.

Figure 2, Rothwell and Zegweld, interactive model of innovation [RZ85]

New products are influenced by their developers, existing or potential customers, marketplace conditions, available technology and needs of the whole society. In his classic book "Discipline of Innovation", Peter Drucker stated: "it (innovation) requires ingenuity and focus". Neither an idea, invention or process can became a thriving success unless it is clearly outlined, implemented and and widely utilized [dis85]. Wide utilization requires differentiation, customer acceptance, seamless delivery and a sustainable business model [Tro08].

According to Poppendieck and Poppendieck, software development is a special form of new product development [PP07], p.17. Therefore, software development efforts make no exception: to succeed, a software product needs acceptance from it’s stakeholders such as business owners, potential customers and end-users. It has to be made available for delivery, differentiate from it’s competitors, and it needs to be
built on a sustainable business model and an architecture that meets the needs of the stakeholders also in the future.

**Open Innovation**

New product development was long based on secrecy of design and stealthiness in between the development organizations, leading to creating a network of legislation for strict proprietary rights and patents. A recent theory of Open Innovation, on the contrary, emphasizes the value of seeing one’s network as an asset for productive influences and collaborative creativity [CVW05].

"Open innovation is the use of purposive inflows and outflows of knowledge to accelerate internal innovation, and expand the markets for external use of innovation, respectively [CVW05]."

The open source -community has emerged as a manifestation of the open innovation paradigm. In 2013, software architectures and applications are created with tools and services that are developed and maintained by others. By combining professional expertise across organizational borders, new opportunities and faster time-to-market can be expected [GEC10]. The ever-expanding options in software construction puts technical experts in a key role in communicating the limitations and possibilities of the technical solution space. Also the ability to recognize and e.g. different architectural decisions’ effect on the cost structure of the software product is needed. Equally important for understanding non-functional requirements is to comprehend and predict stakeholders’ requirements for future use of the software.

Nambisan and Wilemon concluded in their survey that the development process often benefits from emphasizing the multidisciplinary project initiation, allowing the team to be better equipped for responding to changes in e.g. business conditions during the development effort [NW00]. As agile software development methods emphasize close and frequent interaction in between the development team and it’s stakeholders, the ability to work collaboratively with people from varying backgrounds and professional disciplines has gained value in the repertoire of a software
developer.

The End-User

Before the discovery of the open innovation-paradigm, innovativeness was long seen as an asset of product development organizations. The debate about whether the best ideas originate from marketing management visionaries, research and development engineers or hands-on manufacturers was thus stirred by Eric Hippel's "lead-user innovation"-theory published in 1986 [VH86]. Hippel summarized the twofold relationship in between the producer and consumer as:

"Manufacturers possess the knowledge of the solution possibilities, while users possess knowledge about needs [VH86]"

He recommended that new product developers should abandon their costly striving to understand customer needs, motivations and dividing segments through market research and ethnographic studies and instead, let users express their desires by building prototypes within a solution space that has been fitted to the problem complexity and capabilities of the problem-solver [VHK02].

In software development, new ideas are often created co-operatively with end-users [PCBH11], p.79. Various design methodologies emphasize including end-users into the design process [HWW05]. Usefulness, acceptance and success of software products are believed to be the result of involving end-users deeply into the design process [Tro08, HWW05, PP07, PCBH11, VH86]. The end-user paradigm is emphasized also in New Product Development praxis, where discovering customer segments, archetypical customers and their needs has become the starting point of product design efforts [OP13, Mau12].

In the increasingly complex market of products and services, the diversity of user needs is greater than ever before. The process of entering the software market has become easy, making the competitive environment even more aggressive and prone to rapid change. Meanwhile, battle for customer attention has become increasingly difficult [PCBH11] p.20-41. Corporations do not only compete with their peer rivals,
but also with highly skilled and agile individual innovators [BVH09]

### 2.2 Uncertainty in Software Projects

Understanding sources and nature of uncertainty in project front-end is beneficial for the development effort [KW02]. Pearson (1991) described in his article "Managing Innovation: An Uncertainty Reduction Process" a vehicle for understanding and classifying innovation projects or vague ideas on their level of uncertainty [Pea91]. The framework separates issues concerning output of the project ("what is to be done") from the development process ("how it is to be done"), revealing four categories of different kinds of uncertainty. The "Pearson's Uncertainty Map" is illustrated in Figure 3. In [PCBH11] Pikkarainen et al. applied the map on software development and identified recommended practices for each of type of the four. The following introduction describes both: the nature and appropriate means for managing uncertainty in deciding software project priorities.

![Figure 3, Pearson’s uncertainty map [Pea91].](image)

**Combining market opportunities with technical capabilities (4)**
Software ideas that can be positioned to the fourth quadrant usually represent core growth opportunities for a business: creating new products with existing technology [Tro08], p.86 or developing existing products further incrementally [PCBH11], p.56. Uncertainties about "whats" and "hows" are low, giving the emerging development effort a clear vision and scope.

Product ideas in this quadrant can be evaluated by setting e.g. a quantitative feasibility-, cost/benefit- or urgency criteria and ranking them accordingly. Usually only 20 per cent of ideas are found feasible or promising in the first screenings. Therefore, while working with core growth -ideas, it is important to find a prioritization method that can evaluate a big amount of ideas in a short time [PCBH11], p.56.

**Application engineering (3)**
Third quadrant includes classical cases of technology push: ideas that deal with discovering new uses for an existing technology, i.e. problems to a solution that is at hand. The means of implementation is easily identifiable: e.g. the Application Programmer Interfaces may already be available.

**Development Engineering (2)**
Second quadrant includes ideas that may have a clear goal: a commercial opportunity may have been spotted and the product idea has been crafted accordingly. Ideas belonging to this quadrant represent the classic "market push" -approach to new product development. Software developers have more freedom in choosing the means for implementation, but a fixed goal gives a less turbulent start for the development effort.

Straight forward quantitative assessment is not applicable for either Application-(3) or Development Engineering (2) ideas. To avoid early termination of promising software products, further development of the idea is necessary. Instead, categorizing and clustering the ideas may be beneficial in discovering the radical ideas that posses an opportunity for people-centric, collaborative development [PCBH11], p.p. 58-59.

**Exploratory Research (1)** When uncertainty about both the process and the means is at highest, exploratory research (1) ideas take place. This includes situ-
ations when both the goal of the project and the technology applied are not understood. Analyzing the possibilities and shaping research ideas is highly domain-knowledge dependent and therefore, no universal method exists for evaluating the feasibility of ideas belonging to the category [PCBH11], p.60.

To conclude, the less knowledge about the project goal or means to achieve it, the more uncertainty is involved in the development process and the bigger risk a software project failure is. Poppendieck and Poppendieck state in [PP07], p.29 that the software development effort is in fact, a process of creating knowledge. Reducing the uncertainty about the process decreases over the time while development team learns about the problem- and solution space they are working with. Uncertainties about the project’s outcome can be helped by aligning software development team’s work with stakeholder goals and prioritizing development tasks accordingly [Pic10], p.24, [KS08], p.105.

2.3 Software Project Success

Success of software development efforts have long been contemplated by both practitioners and researchers. Exhaustive quality models have been created to quantify software development excellence and frameworks for quality management and measurement, such as GQM, CMMI, QFD have emerged.

According to one of the classical definitions, a software project’s success depends on performance of project management and end-product acceptance [Bac99]. Project management success is indicated by timeliness, cost and quality of the product [Atk99], while end-product acceptance is characterized as:

"(1) The degree to which a system, component, or process meets specified requirements." [IEEE 610.12-1990]

To ensure customer acceptance, methods for managing the development by customer-written acceptance tests, such as Acceptance Test Driven Development [Gär12] have emerged. IEEE standard 610.12-1990 further elaborates the quality of customer acceptance as:
"(2) The degree to which a system, component, or process meets customer or user needs or expectations." [IEEE 610.12-1990]

Stakeholders do not intrinsically accept a software for its technical excellence, reliability or novelty, but for the benefit the end-product delivers [PP07], p.15-17. End-product acceptance is judged by the subjective value that is created to stakeholders compared to their needs, expectations and competitive solutions [Tro08].

Software development practices have long been steered by on risk-driven or value-neutral decision-making. Boehm presented the concept of Value Based Software Engineering in his 2003 article, arguing that when all requirements are treated as equally important, the software product fails to deliver sufficient feedback about usefulness, utility and value to its stakeholders. By measuring the success of a project's "earned value" is with traditional project performance metrics: cost and schedule, the measurability of project performance becomes compromised [Boe03]. The emphasis of requirements engineering, and defining it's success should be moved to fulfilling the real needs of the business- and end-user stakeholders [Orr04]. Table 1 presents examples of stakeholder value.
<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic value</td>
<td>Serving the strategic objectives of the organization, e.g. partnering [ASC07].</td>
</tr>
<tr>
<td>Business value</td>
<td>New products, improving cash flow, higher return of interest [Pic10] p.29, optimized value stream [PP07], p.83.</td>
</tr>
<tr>
<td>Marketing value</td>
<td>Attention and recognition of the product in markets [ASC07], new customer acquisition, loyalty from existing customers.</td>
</tr>
<tr>
<td>End-user value</td>
<td>New service, facilitation of tasks, ease of use, utility, consumability [KS08].</td>
</tr>
<tr>
<td>Business customer satisfaction</td>
<td>e.g. renewal of contracts, product recommendations to other customers [ASC07]</td>
</tr>
<tr>
<td>Sales value</td>
<td>Adding to the possibility for increased sales [ASC07]</td>
</tr>
</tbody>
</table>

Table 1: Examples of value for different types of software project stakeholders.

Poppendieck and Poppendieck consider features that serve neither meaning or value to stakeholders as being the worst kind of waste of resources in software development [PP07], p.25. Disregarding business- and end-user needs may result in low business performance of the product [PP07]. When striving to capture the most essential software process from the viewpoint of the end-user and prioritizing it the highest, the first product release should contain least risk to be rejected later during the development process [Boe03]. Finding the most critical features to begin the development effort with ensures that the "system’s core purpose" becomes top priority [Orr04]. Boehm’s and Orr’s ideas are reflected in practitioner books, e.g. IBM’s Outside In Development [KS08], which is referenced later for describing practical approaches on value- and critical process oriented requirements engineering.

To summarize, traditional software project- and product metrics do not provide sufficient guidance for agile software development. In the dynamic and unpredictable context software development efforts take place in, the focus of software companies has shifted from engineering-driven practises to innovation-driven development. Defining success of software development efforts by traditional metrics is not enough for success [PCBH11], p. 193, [Orr04].
2.3.1 Project Front-End

A software project front-end, or in IEEE’s terms, "concept phase", is described as being a period of time during which the whole project or a development cycle is initiated. During that period the user needs are assessed, described and documented [IEEE s 1002-1987 [91] and 1012-1986 [121]]. The definition is further augmented by The Software Engineering Body of Knowledge by describing the initial phase of a project being the time when also planning of the project’s activities and defining the project scope occur [BRL+02].

Kim and Wilemon define a new product development "front-end" as a process starting from discovery of a promising idea to finishing it as a specification, fit for commencing the development effort [KW02]. Similarly, the goal of a software project front-end is to choose what exactly is to be built and create a definite enough specification needed to start development [Bro87].

As one-sentenced descriptions fail to capture the variability in software project initialization, [KW02] further specify qualitative factors that differentiate the new NPD front-end from the development phase. Table 2 summarizes the differences.
### Table 2, Comparison between the front-end and development phases [KW02].

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

The front-end phase in both New Product Development and Software Development is unstructured and hard to measure. Nonformity of ideas prevents starting the development effort [KW02]. Success and performance of pre-development activities rely on the team’s ability to reduce uncertainty during the project front-end [KW02].

In their book ‘Lean, Rapid and Profitable New Product Development’ Cooper and Edgett outline fundamental mistakes in introducing a new product that are observed to lead to low product performance [Tro08]. Table 3 describes the most common problems resulting from insufficient development project front-end work.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Missing of a compelling value proposition</td>
<td>Lack of customer interest</td>
</tr>
<tr>
<td>2. Neglect of defining a competitive advantage</td>
<td>Poor differentiation</td>
</tr>
<tr>
<td>3. Insufficient innovation front-end work</td>
<td>Slowed development due vision in clarity</td>
</tr>
<tr>
<td>4. Insufficient voice-of-customer evaluation</td>
<td>Poor user acceptance</td>
</tr>
<tr>
<td>5. Lack of end-user validation of concept</td>
<td>Poor user acceptance</td>
</tr>
<tr>
<td>6. Changes in product specifications</td>
<td>Delays due to misunderstandings</td>
</tr>
<tr>
<td>7. Variance in scope</td>
<td>Difficulty of estimating amount of work</td>
</tr>
<tr>
<td>8. Lack of cross-functionality in team</td>
<td>Decreased team cohesiveness</td>
</tr>
<tr>
<td>9. Under-resourcing the team</td>
<td>Delays due task switching</td>
</tr>
<tr>
<td>10. Lack of competencies</td>
<td>Inability to proceed with the project</td>
</tr>
</tbody>
</table>

Table 3: Reasons for low product performance [PP07].

To conclude, starting a software project with sufficient front-end work decreases uncertainty about the project’s goals. Well-defined project goals and scope create a realistic ground for estimating the costs of the software development effort.

### 2.3.2 Initial Product Planning

The later changes to requirements or acceptance criteria of features occur, the more costly a development effort becomes [TF00]. Therefore, careful assessment and validation of ideas is required especially in initial product planning. A Product Owner is responsible for creating and maintaining a statement of the overarching goal and reason for the development effort’s existence, the Product Vision [Sho08, Sch09]. He should gather goals and requirements from stakeholders and negotiate compromises on what features to include in the software product and what to leave out [Pic10]. Both XP [Warden2008], Scrum Project- [Sch09] and agile Product Management [Pichler11] recognize the Product Owner having a critical role in linking business stakeholders’ requirements to software feature prioritization. Neither XP
nor Scrum methodology do specifically address the requirements engineering, analysis and specification phases [Sho08, Sch09]. Instead, the Product Owner is expected to formulate a plan for doing so [Pic10].

A new software development effort often starts with ideas that are focused on the project deliverables [Sho08]. At best, decisions made in the initial Product Vision balance between stakeholder value, such as sales opportunities, differentiation in the marketplace, technical competencies, architecture, quality requirements and long-term product planning. The crux of successful product management as being able to distinguish and balance in between commercial and customer value, while maintaining a vision of the product’s long-term development plans [Sho08, Pic10]. However, the ability to differentiate individual stakeholder value from the surge of ideas is complicated [TCBB09]. In practice, to avoid over-specification, stakeholder prioritization and value analysis can be neglected or executed only partially.

Results of the project front-end elicitation and analysis is captured in a product backlog as user stories [Pic10, Amb02, Sch09]. Stakeholder priorities are reflected in the order of the product backlog items. The Product Owner divides the Product Backlog into proposed swatches, including sections for e.g. sprint-, release- [Pic10], MVP-, and long-term product planning. This is the software development project’s starting point and the contents, priorities and grouping of the Product Backlog are expected to evolve ever since the development project starts. The Product Backlog is influenced by also initial architecture decisions and on how quickly or slowly the software development team can transform the Product Backlog items into functionality as soon as the first sprint is planned [Sch09, Pic10].

2.3.3 Long Term Product Planning

Software development plans encounter perpetual need for change while Product Owner and the development team learn about technical solution space and stakeholder needs. Flexibility from the development process is required [Tro08], p.73 [Sch09] and software community has answered to the need by introducing agile: iterative and incremental work processes that allow fast reacting to changing requirements, experimentation and learning by doing. Iterativity helps projects where no exact up-front specification can be formed to react to changes in priorities and adjust project plans accordingly [Tro08], s.59.
Jarmo Vähäniitty presented in his PhD thesis [Väh12] a comprehensive model for the cycles of long-term product planning in the Agile Software Development context. Vähäniitty's work is based on Cohn's "The Planning Onion" -model [coh06] and augmented with Smits' model of agile planning artefacts [Smi07]. The model is presented in Figure 4.

![The Planning Onion](image)

Figure 4: The Planning Onion [Väh12].

A strategy is a long-term action plan for an organization. Portfolio planning reflects the company strategy in managing a set of products (i.e. the Portfolio). A Product Roadmap captures the high-level goals of the future product, which reflect the project’s stakeholders’ needs, goals and expectations [Pic10]. It is a view on the product backlog’s coarse grained items [Väh12], p.35 that according to [Pic10], should be constructed after the release of the Minimum Viable Product.

A release plan is an outline how the project should proceed during next iterations.
It communicates the product strategy internally, facilitating communication in between the product owner and the software development team [Pic10]. Release planning is an ongoing task and scope of a release is prone to change. A plan of deciding which features the next release should contain should span only to the immediate launch of the software product. Deciding on which features the next delivery is very complicated, or even impossible for a single product owner due to the difficulty of quantifying value of the features. According to Tourwe et al, practical approaches for defining a release scope are ad-hock. No definition exists on what work-steps planning a release should include [TCBB09].

Release planning provides information allocating the budget and resources for the project: a set of functionality, accompanied with cost and time for bringing them to life [Väh12] p.35. Planning a release and estimating project costs are easy when uncertainty about the project’s goals is low [TCBB09]. However, this is rarely the case.

An iteration is a timeboxed period that is planned in advance to contain features from the product backlog that should be developed as soon as the software team is ready to implement them [Pic10], p. 48. Goals of each iteration are captured in a sprint backlog with fine-grained items that are ready to be specified with more detail in e.g. acceptance tests or implemented as is. The release plan guides Sprint Backlog prioritization, also essentially effecting definitions of the goals of each sprint. The agile development team should contribute in choosing the sprint goals during sprint planning sessions. Daily tasks are selected from the sprint backlog by the self-organizing software development team to be designed and implemented on a daily basis [Sch09, Pic10], p.59.

While the emphasis of agile methods are focused on a single development team working on a single project [KL08] and close collaboration of the development team and it’s customer, they lack support for pre-project activities that are essential to new product development. The knowledge gap created in between long-term planning and daily activities is reported to distract, even to disable determined product development [DFGSO10].
2.4 The Software Product Vision

The overriding reason for communication problems in software requirements engineering lies in the fact that software design and development is very much a behavioral process, where human and organizational elements have an important impact on the design [CM02]. In a small example project organization of three people: software customer, developer and end-user, all parties possess individual tacit knowledge about their domain [NT96]. The knowledge may include for example knowledge about underlying business processes, end-user work processes or opportunities and restrictions that arise from technological range of options. During requirements elicitation the goal and details of the development effort are discussed. A first impression on shared knowledge (x,y,z) is acquired. While the work proceeds, the understanding grows. The amount of mutually shared knowledge (S) grows into a converted, commonly agreed mental model about the project’s goals [NT96].

Figure 5, Visualization of tacit, explicit and converted organizational knowledge applied to an example software project organization.
If unarticulated, the shared knowledge (S) is only seemingly unanimous [NT96]. Stakeholders evaluate the same information need differently, or may see requirements as raising either conflicting or competing use of limited resources, according to their role in the organization, background or mindset. By externalizing the knowledge into a verbalized form, and combining it to a coherent, articulated whole, a shared mental model (S) can be created [NT96].

2.4.1 Product Envisioning Benefits

Wilemon et al. concluded in their survey that a software development effort’s overall success and profitability benefits from a clear initial product concept, sound scope and definitely specified critical features that serve subjective and genuine value to all stakeholders [KW02]. However, the concept of a Product Vision has traditionally been present only at levels of company strategy- and product portfolio management [Väh12]. It has been recognized that agile development methods should be extended by tools for long term software product evolution and portfolio management [KL08] and methods for tracking the high-level product concept and underlying goals have started to emerge [Väh12, Pic10, Pic13].

A Product Vision describes the high-level goal of the development effort, it’s reason for existence [Pic10, Sch09, Tro08]. A shared goal of the project should be ideally accepted by all project stakeholders. It should become clearer as the project moves forward and the development team learns about for example the problem-, solution-, technology- and market spaces. While being responsible for acquiring funding for the project, the Product Owner is also responsible to delivering the vision in a manner that maximizes their return on investment for the business stakeholders [Pic10], p.25-29.

The greater amount of uncertainty there is about the software project’s goals, the bigger risk of failure the project is subject to [Pic10] p.57. Developing vague ideas into software concepts is a knowledge creation processes, which aims to reduce uncertainty until a development plan is discovered [PP07] p. 14-17. Having a tangible Product Vision facilitates communicating with stakeholders and thus, determination of the Minimum Viable Product (MVP) as well as setting priorities for the development effort aligned to it’s business goals [Pic10], p.24,28,53.
While stakeholders are eager to share their needs and aspirations, goal of the software project may become fuzzy. Backlogs or issue trackers are filled with features, restricting the ideas becomes essential. Defining an innovation target, such as a Product Vision, filters out the ideas that really make a difference for the project and it’s stakeholders. Clear goals and well-defined targets also enhance creativity while all stakeholders are able to focus their activities, yielding more concrete suggestions and better quality of development ideas [PCBH11], p.27. As to create a Product Vision, it is equally important to outline the actions that need to be taken in order to reach the desired state [PCBH11] p.39.

Based on the initial Product Vision, Product Owner and the development team capture stakeholder needs and make assumptions on the future use of the software, creating interpretations on non-functional requirements such as scalability, changeability and robustness. The early assumptions may effect the baseline architecture of the software significantly. To make informed decisions, the software team should be aware of also the maturity of the Product Vision, helping to predict changes in the development plans.

2.4.2 Product Envisioning Problems

For both the development team and product owner, it is meaningful to know what the development effort is heading towards, but equally important is to identify the directions the effort shouldn’t take. A clear vision statement helps the software development team to distinguish the features that are the most critical the to building the product [Boe03]. Common mistakes in creating a Product Vision include:

No vision
Development starts with customer describing software features. Stakeholders, their needs or critical product features related to the stakeholders’ goals are not addressed or are evaluated superficially [Pic10]. No software quality requirements or baseline architecture goals are addressed before starting development [?, KS08]. Insufficient front-end work contributes to vision inclarity, which increases the overall cost of the innovation effort by creating misunderstandings, slowed development due changes
End-user knows it all [CE09] 
Following the customer’s needs without a Product Vision will unlikely lead to a successful, differentiable product.

Prophecy vision [Pic10] 
A fantasy of the product which contrasts with the development team’s ability to produce software. While provides a valuable learning experience for the customer, may lead to exhaustion of the development team.

Scope creep 
Changing product specification and project scope are often created by product insufficient front-end work [Tro08] s.7. Frequent changes in the scope of the development effort results in difficulty of estimating the real amount of work, thus leading to difficulty in project planning.

Requirements churn 
Changing priority of features creates confusion amongst the development team. Goal of the project becomes fuzzy, effecting intra-project prioritization. Requirements traceability is complicated due to incomplete understanding of connections in between stakeholders and requirements [KS08] p.20. Delays due to misunderstandings occur, increasing the communication costs of the development effort [PP07].

"We know what is good for our customers" 
Insufficient voice-of-customer work in product design contributes to low customer acceptance [PP07, Pic10]. Establishing software design on unvalidated assumptions about the end-user or letting technological solutions shape the product risks the project creating a software that nobody wants [KS08, PCBH11] p.41. Neglecting validation of the product concept or features with real end-users contributes to poor user acceptance [PP07].

Big is beautiful, Big Up-Front Design, Big-Bang Development, Analysis paralysis [Pic10]
Investing time and money in specifying a software too completely in advance [Pic10]. While reaching for a complete understanding of the end-user, market or technology, the front-end work is overdone. In all cases there is a high risk of failure due to impaired ability to react to changes during development [KS08]. Cooper and Edgett encourage an innovator to identify only the most important decisions and key processes that are needed to be specified in more detail [Tro08], s.56. Boehm recommends to identify only the most critical process of the system and start building the software iteratively around it [Boe03].

Vision not Properly Translated into usable Guidelines [?, ?] If no vision exists, the development team may try to determine how the big company vision should be interpreted as a practical level to form usable guidelines for development. Creates confusion in e.g. prioritizing tasks or product features [PCBH11] s.42. Collaboration between stakeholders is complicated due to misunderstandings.

According to Poppendieck and Poppendieck in their book "Lean and Profitable Software Product Development", a vision statement strives to describe clear a outline of the product to be developed, while leaving room for learning whilst in development. A software vision should not contain functional features, but bird’s eye view of the critical success factors in [Pic10], p.27. Externalizing the assumptions help e.g. product owners to iterate, test and refine their ideas with stakeholders. Tacit knowledge should be made visible as a one-page summary that explicitly displays the whole concept [PP07], s.13.

Software Product Vision models presented in this chapter include short descriptions from various authors (D), Geoffrey Moore’s definition from his classic book "Crossing the Chasm" (1991) in [Moo91] (E) and a more recent version from agile product ownership specialist Roman Pichler, who presents a two-stage envisioning model: the Product Vision Board (F) and Product Canvas (G) [Pic13].

2.4.3 Established Product Vision Models

Product vision is a concept that has long been present in New Product Development literature. Also in Agile Software Development literature, various short descriptions of Product Vision statements have appeared. [Hoh07] recommended imaging a box
where the product should be shipped in. Later, [Hig09] p.96-97 recommended to form the vision in three concise bullet point descriptions of what would sell the product. An alternative take on Product Vision can be found from [coh06] p., 232, who instructs the team to imagine how the product would be reviewed after launch. Leffinger describes the Product Vision consisting of a collection of epic user stories [Lef10]. All short vision statements focus on the deliverables of the software project and do not assist in e.g. defining stakeholder value for the Product Vision. Therefore, they were not included in semantic analysis of Product Vision statements described in chapter 2.5.

In this chapter, three structured Product Vision statements are presented: Cooper et al. Integrated Product Description (A) Geoffrey Moore’s Product Vision Statement (B) , Alex Osterweilder’s Business Model Canvas (C) and Ash Mayura’s Lean Canvas (D). In addition, a two-step, experimental Product Vision model (E and F) is presented in chapter 2.4.3.

A. Cooper et al. - The Integrated Product Description (IPD) [Tro08]

In his book "Lean and Rapid New Product Development", Cooper et al. summarize a high-level product definition from the viewpoint of product innovation management. The Integrated Product Description (IPD) is based on the author’s practitioner experience in developing the Stage-Gate process. The IPD suggests a structured model for a Product Vision [Tro08] p.56 that is not accompanied with either heuristic for the envisioning process or instructions on what to include in each parameter, leaving room for generous interpretation. The product description consists of definitions of:

- project scope
- target market
- product concept
- positioning, e.g. price target
- value proposition
- benefits to be delivered
- features, attributes, requirements
- high-level specs

Even though the model provides a generous amount of variables that can be specified in great detail (e.g. "features, attributes, requirements" and "high-level specs"), the Cooper et al. emphasize that a light-touch iterative development of the model should be preferred instead of striving to describe the whole product up-front. While information about the problem domain becomes less uncertain, the model may be iterated further or a separate variation may be created easily to allow e.g. concurrent design of the product concepts.

B. Geoffrey More - Crossing the Chasm [Moo91]

In his classic reference book for technology marketers, Geoffrey More described that a holistic Product Vision could be communicated with a structured template, consisting of the following properties:

- For (target customer)
- Who (statement of the need or opportunity)
- The (product name) is a (product category)
- That (key benefit, compelling reason to buy)
- Unlike (primary competitive alternative)
- Our product (statement of primary differentiation)

Moore’s software Product Vision statement has become a widely utilized means for expressing the software product’s high-level goals and it is referenced often in both practitioner and research literature.

C. Alexander Osterwalder - The Business Model Canvas (BMC) [OP13]

The Business Model Canvas is a light-touch approach for Product Vision that pays special attention on defining the business logic of a new product in the form of a single page canvas. The model is fully applicable to describing a software product. As Cooper’s Integrated Product Description, also the Business Model Canvas
is designed for iterative and parallel development of concepts. Figure 6 depicts the Business Model Canvas.

![The Business Model Canvas](image)

Figure 6, Business Model Canvas -poster [OP13]

The Business Model Canvas consists of 9 elements, the application of which is guided by questions and examples attached to each element. The model is encouraged to be applied in a sequential design process. In this introduction, the canvas elements are presented in the order of the heuristic.

1. **Customer Segments**
A customer segment describes a rough caricature of the business’s customers based on their needs or demographic attributes. The customer segment can be refined by performing a customer discovery process, which will produce archetypical end-user or customer descriptions, personas [Amb02, OP13].

2. **Value Proposition**
A value proposition describes the genuine benefit or value that the product will deliver to its stakeholders. A value proposition may be qualitative (best service) or
3. (Delivery) Channels
In order to reach it’s customers, the product must identify the channels through which it’s will be delivered to it’s customers. In case of software development, the delivery channels may be e.g. different end-appliances or strategy for software dissemination (e.g. iTunes, open source).

4. Customer Relationships
Nature of interaction with the customer is described in the Customer Relationships -section. The strategy for interaction may be e.g. co-creation of the product or service with lead users: building a community of independent developers to support open software innovation.

5. Revenue Streams
The Revenue streams -section defines means the product should use to yield income from it’s customers, partners or other internal or external stakeholders. In software products, different revenue stream models can be e.g. subscription, licensing, micropayments or advertising.

6. Key Resources
When building a product, a basic amount of resources are needed to build, launch and support the product. Archetypical resources for software development are devices, software platforms and storage media, the deployment of which has been facilitated by cloud based services’ becoming common.

7. Key Activities
Key activities describe the most important processes that establish the product. All activities envisioned should enable fulfilling the product’s value proposition defined in step 2.

8. Key Partners
Key Partners include stakeholders that help to bring alive and sustain the product’s operation. Choosing partners may be motivated by risk reduction, mutual benefit
or in the case of software development, e.g. streamlining operational tasks by outsourcing IT infrastructure maintenance.

9. Cost Structure
Cost structure describes spendings of the business, such as service- or maintenance fees, costs for delivery etc.

Prioritization of stakeholders and software features is guided with questions in (delivery) Channels and Customer Relationships -sections. Both questions emphasize analysis on cost-effectiveness of the solution.

Ash Mayura’s Lean Canvas approach has been created an amendment of the Business Model Canvas, that takes a more concrete approach in describing details of the product design, "modeling the runway to your MVP". Mayura provides a wide set of advice on how to work with the canvas, giving depth to the design process. However, only a small fraction of the instructions are visible in the canvas tool itself, as seen in Figure 7. Similar to the BMC, also the Lean Canvas has a heuristic that is recommended to be used in design. This introduction follows the predetermined design process.
1. Problem
Mayura’s primary assumption is that a product should be based on a real problem or friction that a wide set of possible end-users/customers experience. In his book, Mayura encourages comparison with existing means in solving the same problem.

2. Customer Segments
The heuristics for defining the problem include also stakeholder analysis and customer discovery - identifying user roles and distinguishing especially early adopters in the target audience.

3. Unique Value Proposition
A unique value proposition captures a clear and compelling promise that states why the product is "different from it’s competitors and worth paying for". The analysis evokes performing competitive analysis.
4. Solution
Sections 1-2 describe the product features from the light of the customer. When analyzing the solution, the product is sketched with a light touch. Detailed planning should be avoided and the final concept or design should be, in Mayura’s words, be discovered as late as possible.

5. (Delivery) Channels
Different paths of the product to it’s customer is likely to be discovered as a byproduct when analyzing sections 1-3. The Delivery Channels -analysis should include a light cost/benefit analysis and contemplation on the nature of the channel (in/outbound, direct/indirect/automated) to discover which product lifecycle state the delivery channel can be applied to. Delivery Channel-analysis helps defining cost structure (5) of the product.

6. Cost Structure
In final, financial analysis, cost structure of the product is sketched. Identically to Osterwalder’s cost structure (9.), outflow of money can include resources, delivery channel costs or maintenance fees.

7. Revenue Streams
The business model’s revenue strategy is initially chosen, identically to Osterwalder’s canvas (5).

8. Key Metrics
Key metrics describe which activities are required in order to keep the product at a sustainable level of operation. Metrics may include e.g. financial targets, such as a desired amount of paying customers, chargeable transactions or acquired visibility in a marketing channel. In case of software products, the Key Metrics give valuable input on nonfunctional requirements of the product.

9. Unfair Competitive Advantage
To ensure differentiation, Mayura points out the significance of defining a competitive advantage. In Mayura’s words, it is the hardest of product formation problems.
2.4.4 Experimental Product Vision Models

The Agile Development community has only recently realized the power of creating a Product Vision for focused development. Much of the sources for information on building a software Product Vision are unfortunately still corporate white papers, hands-on practitioner books or blog posts. While some research on bridging the gap in between product management, software development and user experience design exists [Moo91, HKRM09, SM03, LKV07, Väh12], the topic of initial envisioning with the help of a structured Product Vision and its longitudinal effects of project- and end-product performance remains scarce, if nonexistent.

Roman Pichler has recently elaborated Moore’s concept towards the direction of end-user- and market driven software product development. In October 12, he published a blog post presenting a practitioner’s approach for software Product Vision. Even though the model in this thesis was created independently and prior to Pichler’s Product Vision framework, his contribution is presented for underlaying relevance and validity of the Product Vision framework presented chapter 4. Pichler’s Product Vision framework consists of two tools: the Product Vision Board (E) and Product Canvas (F).

E. Roman Pichler - The Product Vision Board (P.PVB) [Pic13]

Pichler’s five-parameter Product Vision Board for captures elementary assumptions about the software product to be built and is intended to be used by the product owner to initiate software product design. Figure 8 displays the P.PVB. The product Vision board itself does not provide guiding questions. The vision board is accompanied by a recommendation for it to be applied from left to right. This introduction follows the design heuristic.
The product Vision Board consists of:

1. **Vision Statement**
   A short sentence describing the very broad and high-level goal of the project. In later instructions, Pichler identified the "Vision Statement" to be identical to "Value Proposition".

2. **Target Group**
   A list of stakeholders: who should use the product and who would purchase it.

3. **Needs**
   Pichler encourages finding a clear definition of the problem that is to be solved. He also emphasizes the significance of analyzing how well the designed solution addresses the discovered needs. Technological, legislative constraints are also encouraged to be identified, producing non-functional requirements for the analysis. However, doing so is not instructed in the canvas tool itself, which may cause the analysis to be only partial.

4. **Product**
   Pichler instructs to describe only 3-5 top product features, prioritized so that the top feature satisfies the top need of the end-users or customers. In his blog, he instructs to innovator to identify a feature that the product can’t survive without,
similar to [Boe03].

5. (Business) Value
In the last section Pichler encourages analysis of the goals for the product in regards to it’s business owners. He instructs finding revenue streams and identifying the cost structure that effects sustainability of the software product.

The Product Vision canvas is encouraged to be used as an initiation of the product innovation process, rather than a comprehensive and detailed design of a MVP or a design brief for the software development team.

F. Roman Pichler - Product Canvas (P.PC) [Pic13]

Earlier in July in his blog a more detailed and practical Product Canvas: a shared space display poster for an agile software team as a tool for keeping track of Product Vision and facilitating long-term planning of the project’s activities. In the 6-parameter canvas, the target group, needs and the product design solution are described at a more concrete level than in the Product Vision Board. The Product Canvas is intended to assist agile requirements specification in align with the Product Vision Board (C). Figure 9 displays the P.PC.

![Figure 9, Product Vision Board by Roman Pichler [Pic13]](image-url)
Vision and Name
Top row of the product canvas displays a short vision statement and product name. The vision statement should be a short sentence describing the "overarching goal", motivation or the intention of the development effort, identical to a "Unique value proposition". Name of the product can be a sketch or a nickname.

The actual product design is initiated on the second row of the canvas, which consists of three parts:

Personas
While the most important stakeholders have been identified, archetypical characters of users and customers are described in detail with the help of e.g. the Persona-tool [Amb02, SS11].

Journeys
A "Journey" is an interaction in between the end-user and the software. They can be sketched with e.g. customer journey maps, service blueprints, story boards [SS11], usage scenarios or flow diagrams [Amb02].

Epics The column contains "Epic user stories", which are a high-level artifact in the Agile Modeling -methodology [Amb02]. Pichler’s advice is that each epic story should address a specific end-user need.

Design This column is left for sketching the internal software design by e.g. high-level architecture descriptions, user interface flow diagrams or anything that is helpful for the implementation process to start with.

Constraints This section describes the traditional non-functional requirements, according to Pichler, the "operational qualities", which influence the choices of technology and baseline architecture and effect also the cost structure of the end-product.

Ready Stories
The final column of the Product Canvas is dedicated to finalized, implementable user stories [Amb02].

The Vision Board and Product Canvas are developed iteratively throughout the
project, while uncertainty about the problem- and solution spaces decrease. Pichler 
emphasises that Big Up Front Design should be avoided: constant feedback from 
end-users and other important stakeholders will accumulate knowledge about the 
product’s acceptance throughout the development effort [Pic10] p.20,27.

2.5 Comparison of the Product Vision Models

The Product Vision models chosen for comparison shared a common characteristic of all being brief single page tools which can be used for creating new software product ideas. All of the templates encourage iterative development of the concept, enabling also concurrent development of many parallel ideas.

Various short, i.e. one-sentence descriptions describing a high-level Product Vision were found in the literature. However, because they provide very little contribution on analysis of the software idea, they were not included in this comparison. While the Kano-model of product development [SBMH96] is a tool for evaluating product construct and steering the development effort, it was left out of this comparison for it’s narrower emphasis on end-user driven prioritization of product features and attributes. Also, Holtzblatt’s "Contextual Design"-process [HWW05] was first included in the comparison, but later rejected because of it’s heavy emphasis on detailed software design.

Analysis Structure

Because of varying names for coterminous properties of the existing Product Vision properties, a semantic analysis was performed for existing Product Vision models. The Product Vision canvas tools were then studied to find similarities and differences. Assessment criteria for the comparison was: "Does the Product Vision method address this parameter?". The question was answered simply: yes (X) or no ( ). Appendix A describes the mappings in between each parameter and related terms in existing Product Vision templates. 23 parameters were collected from the analysis and organized into five integrative meta-categories (Problem Space, Stakeholders, Solution Space, Market Space and Project Space).
Names of the envisioning methods were abbreviated as follows:

**IPD**: Paul Trott, Integrated Product Definition [?]  
**PV**: Geoffrey More, vision statement [Moo91]  
**BMC**: Alex Osterweilder, Business Model Canvas [OP13]  
**LC**: Ash Mayura, Lean Canvas [Mau12]  
**P.PVB**: Roman Pichler, Product Vision Board [Pic13]  
**P.PC**: Roman Pichler, Product Canvas [Pic13]

**Results**

Results of the analysis are presented in table 4.
<table>
<thead>
<tr>
<th></th>
<th>IPD</th>
<th>PV</th>
<th>BMC</th>
<th>LC</th>
<th>P.PVB</th>
<th>P.P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem Space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem Outline</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem Processes Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td><strong>Stakeholders</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder Discovery</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder Goal Discovery</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Stakeholder Value Analysis</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Stakeholder Prioritization</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Solution Space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution Outline</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution Process Analysis</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value Proposition</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Quality goals</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Requirement Priorities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Delivery Channels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Detailed Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Product Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td><strong>Market Space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competitor Discovery</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competitive Advantage</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Structure</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revenue Streams</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td><strong>Project Space</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Steps</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Project Scope</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Table 4: Comparison Product Vision templates.
2.6 Conclusion

A total of 23 existing Product Vision parameters were extracted from analysis of four established and one experimental Product Vision models. The parameters were divided into five aggregate categories to find common nominators and a structure that facilitated analysis. The analysis discovered that each of the Product Vision models has their own emphasis. In this chapter, the accents and effects of applying the envisioning model in software project front-end are analyzed.

A. Cooper’s Integrated Product Definition (IPD)

Cooper’s eight-parameter Product Vision model bares only very general Product Vision parameters. While capturing a high-level view on the project’s end-product, it does not directly address the problem domain. Surprisingly the model does address the solution space at a very specific level, calling product concept, project scope, high-level specifications and "features, attributes requirements" to be identified. The model pays emphasis on analyzing competing solutions, while paying no attention to discovering end-user or customer needs. It encourages analyzing the benefits to be delivered as well as a value proposition, thus possibly leading the development effort to be based on unvalidated assumptions about the end-user or customer needs. If applied to Software Engineering as is, the Integrated Product Description projects the gap in between new product development and user experience design into the design process, encouraging assumption-driven prioritization of software features in initial product planning.

B. Geoffrey Moore: Product Vision Statement (PVS)

Moore’s 7-parameter Product Vision Statement (1991) is one of the first efforts to specify a software product’s high-level goals holistically. Similarly to Cooper’s Integrated Product Vision, Moore’s vision statement does not encourage continuing problem domain analysis neither to concrete process level. The model does not encourage analysis of the product’s business model but, in align with Cooper and Lean Canvas, emphasizes the value of identifying means for differentiation in the marketplace. However, applying the Moore’s vision statement to software project initiation may cause creating an immature value proposition for the product. Identifying a
compelling value proposition from the viewpoint of e.g. the end-users would require more thorough analysis of the stakeholder’s problem and especially value the proposed solutions provides. If started with PVS, a software project may lack informed analysis of the project space. Scope-, resources and first steps of implementation may become compromised as the model does not address the solution too specifically.

C. Alex Osterweilder: Business Model Canvas (BMC)

When applying the heuristics of the nine-parameter BMC into software product discovery, a strong emphasis is paid on identifying customers or end-users and their needs. The BMC-tool emphasizes defining value propositions and the tone of communication with customers, therefore enabling early concept testing, validated learning and leaving room for end-user innovation already early in the product design process. Analyzing the Channels through which the end-product is delivered to the customers provides concrete input on task prioritization and product’s baseline architecture construction. By analyzing Key Activities the prioritization information is complemented with discovering the most important product features. By identifying Key Resources and Key Partners the aspect of collaborative, open innovation is addressed in the product envisioning process. Outlining the software solution’s key resources can also provide room for initial development- or solution baseline architecture planning. Despite it’s extensive view on intra-product dynamics and the underlying business model, BMC however, does not address the competitive advantage analysis thoroughly. While providing valuable input for feature prioritization, it does not provide input for deciding neither first steps nor the scope of the development effort, leaving the product owner with extensive background information about the product- and business dynamics but little means for prioritizing the development effort’s initial scope.

D. Ash Mayura: Lean Canvas (LC)

Lean Canvas by Ash Mayura is created as an amendment to the Business Model Canvas. The method is described to be the "pathway to Minimum Viable Product", defining an initial product scope with eight parameters. The model has more emphasis on defining a strong competitive advantage than the BMC. Emphasizing
the role of customers and end-users leaves room for validated learning and end-user innovation.

The Lean Canvas is recommended to be applied with a predefined heuristic, which streamlines the design process. Examples from the heuristic include defining an "Unfair Competitive Advantage" before describing the Solution space. Thus product discovery -process is optimized by leaving out design options that create a product that is too similar to the ones present in the marketplace. Starting the business analysis with defining Delivery Channels and their nature and cost structure, the Lean Canvas sets a more solid space for objective business analysis. While being more concrete on specifying the key processes of the product, the Lean Canvas discourages too detailed end-user process analysis. When envisioning Key Metrics for the end-product, analysis of business goals, non-functional requirements and other tangible targets is reflected on the product design. The Lean Canvas leaves definition of first steps, and project resources for development unaddressed. In case of a software project, especially the latter could provide essential input for deciding the product baseline architecture. However, while initial scope of the product release is outlined, it is natural for the detailed analysis and design to be left to the development team and the product owner.

**Roman Pichler’s Work**

Roman Pichler’s two-stage Product Vision system is presented as a continuum from the Business Model Canvas, to an agile software development project, bridging the gap from business strategy analysis to product- and portfolio management and finally to the daily activities of an agile software development effort. Even though the model is informal and not yet published or peer-reviewed in either practitioner literature or research body of knowledge, it’s inclusion for this comparison was essential to reflect on the choises of an expert Product Owner professional on the contribution of the author of this thesis. Pichler’s model was developed independently of this work and published after the author carried out first evaluation cases of the software envisioning framework presented in this thesis.

Pichler’s envisioning system consists of two components: a high level Product Vision Board (P.PVB) and Product Canvas (P.PC) that integrates more closely to the agile
Roman Pichler: Product Vision Board (P.PVB)

The Product Vision Board (P.PVB) pays heavy emphasis on discovering end-users, their needs and the value that the software solution will provide to them. 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 left to further analysis. Evaluating the product’s (business) value is trusted to be made with a more extensive analysis tool, such as a business model canvas. While the Product Vision Board itself does not evoke consideration of means for differentiation in the marketplace, it’s application into the product design process without the help of a separate business modeling tool may lead to a discontinuity of input about existing, similar products and their technical solution space to the development effort. Lack of comparison may result in excluding features and the non-functional requirements of the end-product that could lead to excellence in software product performance.

Roman Pichler: Product Canvas (P.PC)

Pichler’s Product Canvas (P.PC) is a tool for transforming the high-level Product Vision, presented with the P.PVP, into a concrete development plan. It utilizes the Agile Modeling -method’s [Amb02] stepwise refinement of user stories. The canvas has been divided into correspondent sections of Journeys, Epics, Ready Stories and Design. Stakeholder analysis is further sharpened from the stakeholders described in the Product Vision Board. By encouraging the design process into predefined format, the flow of product design is constrained to produce relevant results at each step. Encouraging the modeling of "Journeys", i.e. the user’s interaction with the system in his real-world context, the model may provide more advanced end-user value analysis leading to valuable prioritization information. Consideration of constraints most probably yield relevant information in non-functional requirements for the end-product. However, while modeling the problem- and solution processes, no attention is paid on analyzing the prioritization of the product’s delivery channels, therefore possibly inhibiting informed decision-making about the relevance of each
delivery channel to end-users. While forcing user stories to be made visible the Product Canvas may create overlapping project planning artefacts, as user stories and epics are an essential part of the Product- and Sprint backlogs. However, the combination of Product Vision Board and the Product Canvas may fail to deliver sufficient information for deciding the priorities that contribute to forming the initial product scope. Using the model may also create too detailed specifications of the software too early in the process. Due to it's design, the applicability of Pichler's two-stage Product Vision -method is limited to projects containing human stakeholders. The tool can not be applied e.g. into the design of business process based systems, where interactions can be executed solely in between non-human stakeholders.

2.7 Summary

Uncertainty complicates Software Development. In the large scale, uncertainty may be caused by volatility of the software project’s environment: e.g. changes in the marketplace, societal conditions, legislation or emergence of new technologies. On the software project level, intangible nature of software, abundance of development ideas and varying stakeholder goals create confusion and unsteadiness of the project scope and priorities. Varying competence level of the product owner and software developers may on it’s behalf evoke communication problems: indecisiveness and misunderstandings that may lead to poor project planning and communication breakdowns. Often the cause of uncertainty about project goals originates from participants’ varying backgrounds and unarticulated, tacit domain knowledge.

Effects of uncertainty on project success and end-product acceptance have been widely documented. Uncertainty about the project’s goals and means to achieve them distracts not only daily development activities, software design decisions and project planning but also short- and long-term product development. The amount of uncertainty decreases as the project team and it’s stakeholders learn about the problem-, solution-, stakeholder-, project- and market domains.

By involving the end-user deeply into the design process, a software product’s acceptance and thus, market performance can be enhanced. Analyzing how well the proposed solution creates value for them helps to strive for a better acceptance of
the software product’s users. By assessing competing solutions in the product envisioning process, a means for differentiating the end-product in the marketplace can be identified, again, leading to improved performance. Open innovation can reduce the effort costs by utilizing external resources or establishing collaboration in between organizations.

Sufficient and holistic software project front-end work decreases the impact of uncertainty for the software development project, laying grounds for successful product and product planning. However, the balance in between barely sufficient and over-done project front-end analysis is hard to identify. While striving for reducing the risk of "making the wrong product right", researchers and practitioners have awakened to recognize high-level Product Vision and planning as an important amendment to the agile development process. Benefits of and problems in successful product envisioning exist in both practitioner- and scientific body of knowledge. Some Product Vision frameworks exist in New Product Development body of knowledge and some research on integrating them in agile software development exists. The Software Development community has started to recognize the need for a holistic Product Vision and an Agile Product Management practitioner expert has published his initial, two-step agile Product Vision model. Observing the recent research- and practitioner contributions, it appears that the topic has only started emerging to the awareness of software developers.

When analyzing existing Product Vision models from New Product Development and Software Engineering literature, significant similarities in their underlying principles were found. All models assess some of the same questions with different terminology while some properties are unique for each model. Each of the existing Product Vision model has a strong emphasis on human end-users, which restricts their use in different kinds of software development efforts. Therefore, as a contribution of this thesis, an integrated framework for Software Product Vision is created by analysis of the underlaying knowledge each existing Product Vision model evokes in the design process. In the next chapter, results of the analysis is summarized into a compact Software Product Vision framework.
3 Constructing the new Software Product Vision - framework (InnCa)

Motivation for constructing a new InnCa-framework based on existing vision models was to complement the Software Engineering literature with a new, holistic Product Vision vocabulary. Parameters for the framework were elicited by performing semantic analysis on Product Vision models presented in chapters 2.4.3 and 2.4.4. Goal of the analysis was to find what elementary knowledge each Product Vision model evokes in a design process. 23 parameters from existing models were extracted and organized into five integrative meta-categories.

Problem Space
First meta-category, the problem space, describes the 'as-is' situation without the effect of the software system to be developed.

<table>
<thead>
<tr>
<th>Problem Space</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Outline</td>
<td>Encourages identifying a need or a problem the future software could solve.</td>
</tr>
<tr>
<td>Problem Process Analysis</td>
<td>Encourages to address the process context in which the problem occurs.</td>
</tr>
</tbody>
</table>

Stakeholders

The Stakeholders meta-category contains properties that describe parties involved in creating or solving the problem, including people, organizations and systems.

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder Discovery</td>
<td>Encourages finding stakeholders.</td>
</tr>
<tr>
<td>Stakeholder Goal Discovery</td>
<td>Encourages evaluating intentions and goals of the stakeholders.</td>
</tr>
<tr>
<td>Stakeholder Value Analysis</td>
<td>Encourages analysis of value the system provides it’s stakeholders.</td>
</tr>
<tr>
<td>Stakeholder Prioritization</td>
<td>Encourages finding the most important stakeholders.</td>
</tr>
</tbody>
</table>
Solution Space

The Solution Space -meta-category includes parameters that describe the situation where the product is introduced to the stakeholders for solving the problem.

<table>
<thead>
<tr>
<th>Solution Space</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution Outline</td>
<td>Instructs to outline the solution at a very high level and with a few concise sentences.</td>
</tr>
<tr>
<td>Solution Process Analysis</td>
<td>Encourages analysis of how the solution will interact with it’s stakeholders.</td>
</tr>
<tr>
<td>Value Proposition</td>
<td>Encourages finding a short statement that captures the essential benefits for using the system from the viewpoint of the most important stakeholders.</td>
</tr>
<tr>
<td>Delivery channels</td>
<td>Describes how the product is reached by it’s stakeholders.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Encourages analysis of factors and phenomena that may restrain the project.</td>
</tr>
<tr>
<td>Quality Goals</td>
<td>Evokes analysis of non-functional qualities the software has to meet.</td>
</tr>
<tr>
<td>Requirements Priorities</td>
<td>Encourages prioritization of the discovered requirements.</td>
</tr>
<tr>
<td>Product Name</td>
<td>An initial suggestion for a name of the end-product.</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>The &quot;detailed design&quot; -parameter can be used to identify whether the software project front-end is nearing development phase. It can be also used to control premature design of software features, based on assumptions.</td>
</tr>
</tbody>
</table>
Market Space

Market Space -meta-category describes the product’s relationship to it’s external environment, the marketplace.

<table>
<thead>
<tr>
<th>Market Space</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Competitor Discovery</td>
<td>Encourages analyzing solutions to the same problem that already exists in e.g. the marketplace.</td>
</tr>
<tr>
<td>Competitive Advantage</td>
<td>Encourages to find an unique way to differentiate the product from it’s competitors.</td>
</tr>
<tr>
<td>Cost Structure</td>
<td>Encourages analysis of spendings, caused by e.g. the software’s baseline architecture, possibly encouraging cost-driven project prioritization.</td>
</tr>
<tr>
<td>Revenue Streams</td>
<td>Encourages analysis of the business model, providing business stakeholder goal driven priorities for the development effort.</td>
</tr>
<tr>
<td>Business Performance Metrics</td>
<td>Encourages the business owner to verbalize concrete goals for the software. An example goal may include e.g. &quot;10 000 registered customers per first year&quot;, therefore providing valuable information for software quality goals.</td>
</tr>
</tbody>
</table>

Project Space

The Project Space -meta-category contains parameters that create knowledge that is related to software project planning.

<table>
<thead>
<tr>
<th>Project Space</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>Encourages discovering basic resources that are needed for starting the software project.</td>
</tr>
<tr>
<td>First Steps</td>
<td>Encourages creating prioritized list of concrete tasks tasks that are essential for beginning the software project.</td>
</tr>
<tr>
<td>Project Scope</td>
<td>Addresses the scope of the product by e.g. defining the minimum viable product.</td>
</tr>
</tbody>
</table>
The Software Product Vision

Problem Space
- Problem Outline
- Problem Process Analysis

Stakeholders
- Stakeholder Discovery
- Stakeholder Goal Discovery
- Stakeholder Value Analysis
- Stakeholder Prioritization

Solution Space
- Solution Outline Description
- Value Proposition
- Solution Process Analysis (Functional Requirements)
- Requirement Priorities
- Delivery Channels
- Constraints
- Quality goals
- Detailed Design
- Product Name

Business Space
- Competitor Discovery
- Competitive Advantage
- Revenue Streams
- Cost Structure
- Performance Metrics

Project Space
- Project Resources
- First Steps
- Scope

Figure 10, InnCa-framework.
Figure 10 displays the new, integrated Software Product Vision-framework.

3.1 Product Vision Canvas (InnoCanvas)

The Product Vision framework can be applied by choosing suitable parameters for software project and using it as the form of a simple, canvas-based tool. Figure 11 presents an example canvas tool. The parameters may be complemented with guiding phrases or questions, examples of which are listed in Appendix B.

![InnoCanvas Diagram](image)

Figure 11: An example InnoCanvas with pre-chosen InnCa-framework parameters for a market-driven software product design project.
4 Evaluating the Framework

This section presents three cases where applicability of the InnCa -framework and the related InnoCanvas was evaluated. Participatory Action Research [Why91] was chosen as the research method as the author wanted to maximize personal learning from each case and react to problems arising in the development projects. During all evaluation cases, many rapid and reactive cycles of planning, action, observation and reflection were conducted. Each of the case taught the author different aspects of utilizing the framework.

4.1 Case Selection Strategy

Case A: Software Innovation Workshop
Goal of research was to identify, which Product Vision parameters would be discovered when teaching innovative product design for students of Computer Science as a very rapid workshop course. The research case was planned during March-April and executed in May 12.

Case B: Open data application development
To evaluate the relevance of applying the framework and the canvas tool in a real software development project with moderate uncertainty, a real software development effort with experienced developers was chosen. A startup -style project was executed in September - November 2012.

Case D: Robotics Lab
A project with very high uncertainty about the project’s goals and technologies was chosen to evaluate applicability of the framework in a technology-driven development setting. The project included developing an embedded hardware system and related software. The project started in October 12 and will continue until May 2013.
4.2 Case A:  
Software Innovation Workshop

Software Innovation Workshop was a 17-day workshop course that was arranged at University Of Helsinki’s Department of Computer Science in May 12. Each student was eligible for 3 credit points for passing. Grading of the course was simplistic: passed or rejected.

<table>
<thead>
<tr>
<th>Participants</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11 students</td>
<td>Computer Science majors</td>
</tr>
<tr>
<td>1 facilitator</td>
<td>Author</td>
</tr>
<tr>
<td>3 guest speakers</td>
<td>Industry professionals</td>
</tr>
</tbody>
</table>

During the course, small teams of 3-4 students worked collaboratively to create ideas software products. Schedule of the five workshops was designed to be intensive, consisting of five mandatory 4-hour workshops. In between the workshops, students were allowed to spend 2-4 days contemplating the challenge, interviewing people, validating their assumptions and observing the phenomena related to their tasks. Reports of the extra-workshop activities, as well as final software concepts were delivered as posts and discussions on a course blog. After completing the course, the students were expected to be able to deliver a new high-level Product Vision for a commercial software product as a post-workshop assignment. For passing the course, each student was required to be able to:

- perform a problem-solution discovery by analyzing stakeholders and their needs
- contemplate stakeholder priorities
- identify and validate stakeholder value
- from software feature ideas, identify a critical process
- discover constraints and priorities for a development effort
- create a short and prioritized product backlog [Sch09]

As there are no ‘right’ solutions in new product development, correctness of the students’ solutions could not be verified without creating a prototype and releasing
the it to market. Therefore three industry professionals were invited to talk to allow participants to help define their initial scope and reflect their work on expert opinions and real business practices.

**Action plan**

Two research hypothesis were set before starting the experiment:

A.1 Product Vision framework’s parameters appear spontaneously during a loosely guided software product design process.

A.2 Terminology used in the Product Vision framework is understandable for students of Computer Science.

Using predefined goals for each workshop, the curriculum was designed to conform the principles expressed in the Cooper et al. new product development process "Stage Gate Rapid", which consists of phases of 0) discovery, 1) scoping and 2) building a business case [CE09]. Outline of the workshop themes (WS) and homework (HW) assignments were planned as:

**HW.1** Discover new, exciting technologies.

**WS.1** Perform a Persona interview [Amb02].
- Find a problem to solve.
- Identify other 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 [Boe03].
WS.4 Identify possible partners for providing or consuming the service.
Identify revenue streams.
Create a product backlog.

WS.5 Concept demo

High-level goals of each workshop were introduced at the beginning of each session. Author observed the students’ work process and identified which framework parameters appeared at each stage. Table 5 presents Product Vision parameters as they were discovered for the first time during the workshop. Parameters marked with an asterisk (*) were discovered independent of author’s facilitation.

<table>
<thead>
<tr>
<th></th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-workshop task (HW0)</td>
<td>-</td>
</tr>
<tr>
<td>Workshop 1</td>
<td>Problem outline, Problem Process Analysis*, Stakeholder Discovery, Stakeholder Goal Discovery*, Solution Outline*</td>
</tr>
<tr>
<td>HW1</td>
<td>Competitor Discovery</td>
</tr>
<tr>
<td>Workshop 2</td>
<td>Value Proposition, Stakeholder Value Analysis, Competitive Advantage, Solution Process Analysis*</td>
</tr>
<tr>
<td>Workshop 3</td>
<td>Delivery Channels, Stakeholder Prioritization, Critical Solution Process, Requirement Priorities*</td>
</tr>
<tr>
<td>HW3</td>
<td>First Steps</td>
</tr>
<tr>
<td>Workshop 4</td>
<td>Project Resources (partnering), Revenue Streams</td>
</tr>
<tr>
<td>Workshop 5</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5: Parameters discovered in Case A: The Software Innovation Workshop

Analysis of actions

Most of the work was performed by the students independently. To facilitate tracking the progress, author encouraged the participants to illustrate and record their analysis with any type of method, including classical software modeling tools. Detailed design of the software’s structure was strongly discouraged. Surprisingly, the
students had trouble using modeling tools, such as stakeholder or sequence diagrams to facilitate communication and further analyze their ideas. The modeling tools were used rather to document results, rather than to assist communication during the design process. Discussion of the participants needed to be guided more than was expected, as ideation did not readily provide tangible results.

In the beginning of the project, the students faced maximum uncertainty, as creating a completely new software product concept is an open problem. On the first workshop, after identifying a problem to be solved (problem outline), the students proceeded directly to problem process analysis. Identically, while building a Solution Outline during the second workshop, participants instantly started analyzing the solution processes.

On the third workshop, some participants indicated that they had trouble comprehending the goal of the software product design process. Therefore as an intervention, a 8-segment canvas tool was created to concretize the learning goals. The canvas contained mostly aiding questions that would produce relevant results in reaching learning goals of the course. After introducing the canvas, participants started producing tangible results fast. Discussion about different design options turned into a more concrete level. Figure 11 illustrates the canvas with aiding questions that was introduced to the workshop participants during the third session.

Figure 11: The "product concept" -poster with aiding questions, as presented to
workshop participants.

The canvas tool was originally planned to be introduced at the end of the workshop course to assist a post-workshop product discovery task. It was composed to contain the following InnCa framework parameters:

- Problem Outline
- Solution Outline
- Stakeholder Discovery
- Stakeholder Goal Discovery
- Stakeholder Value Analysis
- Value Proposition
- Solution Outline
- Solution Process Analysis
- Delivery Channels
- Competitor Discovery
- Competitive Advantage
- Revenue Streams
- First Steps (Backlog top 5)

All students were present at the last workshop: the concept demo. Learning goals were validated by assigning an individual product discovery task, which was expected to be delivered in 7 days:

Create a personal software startup idea by going through the design process we used at the workshop. Remember, that the design process shouldn’t be linear. Iterate your concept by what you discover. When sketching your solution, don’t over-analyze. Think about what really is important and what is not. While building the concept, take breaks. Sleep overnight. Talk to people, observe.
Students were provided with both the process description (Appendix C) and the poster (Figure 11) that had been used at the workshop. All participants returned their concepts as requested, yielding a total of 14 well thought, innovative software ideas. Majority of the concepts focused on describing the product concept without too detailed analysis on internal structure of the software. About one third of the concepts were technically very advanced, sketching already a high-level baseline architecture for the software concept.

**Conclusions**

At the beginning of the workshop, it was completely uncertain for the participants where the software product design process would lead. Therefore, Case A is positioned into "exploratory research" in the Pearson’s uncertainty matrix [Pea91]. Hypothesis A.1, "Students discover Product Vision parameters spontaneously", was not valid. The design process performed by novice product developers stalled and needed to be guided with aiding questions about the workshop goals. When participants indicated that the overall goal of the design effort was not clear, a tool for concretizing goal of the product discovery was created as the form of a canvas poster. After the initial confusion, seeing the canvas helped the student to concretize a goal for the software concept discovery effort.

Very few questions about the terminology was asked. Therefore hypotheses A.2 received positive implications: the framework properties were easy to understand for students of Computer Science. Even setting the most intangible goal: an initial "Value Proposition" for the software product was considered easy after accomplishing definitions of the Problem Outline, Solution Outline, Stakeholder Discovery, Stakeholder Goal analysis and Stakeholder Value analysis. A certain amount of design heuristic helped the novice participants to achieve their learning goals.

Some Innovation Workshop participants implied after the course that they had realized for the first time the wider context a software can have. Two examples of final high-level product concepts are presented in Appendix D.
4.3 Case B: 
Open data -application "Taukkis"

<table>
<thead>
<tr>
<th>Participants</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3 programmers</td>
<td>Students, experienced professionals</td>
</tr>
<tr>
<td>1 user experience designer</td>
<td>Industry professional</td>
</tr>
<tr>
<td>1 facilitator</td>
<td>Author</td>
</tr>
</tbody>
</table>

To evaluate relevance of the InnCa-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 software developers assigned for creating an open data application for Apps4Finland -competition. The team consisted of an User Experience Designer (UXD) and three programmers. A goal of the project was to create a winning application and a platform that could be used as the backbone of a software startup.

**Action plan**

Research plan was to identify whether experienced developers could utilize the Product Vision poster that was created in Case A (Software Innovation Workshop). The canvas was to be introduced at the first project meeting.

Research hypothesis:
B.1 A software project front-end with very experienced developers can be facilitated by using a canvas with parameters from the Product Vision framework.

**Analysis of actions**

When starting the project, five parameters of the framework were presented to the development team (Problem, Solution, Stakeholders, Competitive Advantage and Value Proposition). After a bit of a awkward laugh, the developers tried to identify a problem their forthcoming software would solve. The canvas was rejected, thus invalidating the research hypothesis B.1 right in the beginning of the project.

During the first meeting, the UXD presented an initial concept of a mobile, map-
based rest stop application as concept sketch, containing images and user interface mockups. An excerpt of the presentation can be found in Figure 12. After the presentation, author decided to change research strategy and observe which parameters of the framework were present in the UX's presentation and which would appear later during the development effort. Research Hypothesis B.2 was set as "Parameters of the Product Vision Framework can be used to observe the amount of uncertainty about the goals of a software development effort".
After the presentation, first epic user story [Amb02] was identified to be the critical process:

_B.e1:_ "As an end-user sitting in a car, I want to see where I can find interesting places to stop by during my way to my destination."

The development effort started and baseline architecture of the solution was decided by short negotiation in between the development team. The user story B.e1 was divided into smaller tasks. Later, a new epic user story was discovered:

_B.e2:_ As a end-user sitting in a car, I want to see which rest stops are immediately next on my way.

After the first release, which contained both B.e1 and B.e2 the project group decided to deploy an online Kanban-tool for managing development tasks. During the project, development coordination was communicated mostly in online chat and by light user interface sketches made with pen and paper. Author observed and participated in the discussion for the whole duration of the project.

Main functionality, i.e. critical use cases were communicated very early in the project and establishing the platform architecture was accomplished by the programmers communicating directly. Most of the later development ideas were associated with improving the software architecture. Choosing data sources was postponed until the very end of the project. A tool for harvesting user generated content was discussed to be a future development idea. During the project, prioritization of the features or data sources did not follow a need- or value based analysis and was based on availability of the data sources.

After the Apps4Finland voting was over, the software product development effort was discontinued. Software platform created was evaluated by the team to be a valuable asset. For its professional touch and state-of-the-art technologies, the end-product received a lot of attention in the Apps4Finland media near to the end of
the competition. Also, the professionally executed social media promotion campaign yielded many public votes for the competition. Despite its good quality and high rank in the public vote, the competition jury did not choose the application to be one of the finalists.

**Conclusions**

At the beginning of the project, the team had confidence in means to achieve the goal and a clear initial product concept. These conditions place the development effort into the Application Development -category [Pea91]. The User Experience Designer had originally a very clear idea of the software, and communicated it to the development team in the first meeting with his own media. When observing the presentation, many of the framework parameters were present. Only minor adjustments were made to the initial software product concept.

A qualitative study of conception of the Product Vision was conducted as an online form that contained questions evoking the following parameters of the InnCa-framework:

- Problem outline
- Stakeholders
- Solution outline
- Competitive advantage
- Value proposition
- Critical process
- Prioritization criteria
- Next steps

All participants answered the questionnaire with very unanimous answers. Also a quantitative survey on reduction of subjectively perceived uncertainty about the
project’s goals was performed. The survey showed a slight reduction in uncertainty about the project’s goals and means to achieve them during the development effort, giving support for B.2. Appendix ZZZ depict the survey forms. Appendix WWW visualizes results from the survey about perceived uncertainty during the project’s trajectory. Table 6 displays parameters as they were present before, during and after the development effort.

Of the Product Vision Framework parameters Cost Structure and Revenue streams were addressed only superficially. Stakeholder value analysis and defining a strong competitive advantage was also 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, the product did not perform well. When analyzing low product performance with the UXD 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 it’s target audience successfully.

<table>
<thead>
<tr>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Front-end</strong></td>
</tr>
<tr>
<td>Problem Outline, Stakeholder Discovery, Solution Outline, Competitor Discovery, Delivery Channels, Product Name, Critical Process, First Steps</td>
</tr>
<tr>
<td><strong>During Development</strong></td>
</tr>
<tr>
<td>Problem Process Analysis, Stakeholder Goal Discovery, Solution Process Analysis, Value Proposition, Stakeholder Prioritization, Project Resources, Project Scope</td>
</tr>
<tr>
<td><strong>Unanimous answers in the post-development survey</strong></td>
</tr>
<tr>
<td>Product name, Problem Outline, Solution Outline, Stakeholder Discovery, Value Proposition, Competitive Advantage, Prioritization Criteria</td>
</tr>
</tbody>
</table>

Table 6: Parameters discovered in Case B: Open Data Application Development

In the post-project survey, the competitive advantage was identified as being "superior usability". However, it could be identified that the product discovery process lacked a thorough Competitor Discovery and definition of a unique Competitive Advantage. Yet hard to validate, this may also have effected the low product performance. Also, considering Project Resources extensively could have resulted in a
better content provision strategy, e.g. partnering with an existing online community ("Vaihtoehto ABC:lle"). This may have brought access to an extensive end-user community that could sustain collaborative creation of content.

It is impossible to say, whether more thorough analysis of the Product Vision Parameters would have helped to define a software product concept that would have performed better in its competitive environment. However, the InnCa-framework provided in this case, a vehicle for understanding the highly iterative and incremental software product discovery process.
4.4 Case C: Robotics Lab

<table>
<thead>
<tr>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 facilitator</td>
</tr>
<tr>
<td>3 undergraduates of computer science</td>
</tr>
<tr>
<td>1 undergraduate physicist</td>
</tr>
<tr>
<td>Author</td>
</tr>
<tr>
<td>Project Manager, 2 developers</td>
</tr>
</tbody>
</table>

The Robotics Lab was a project for developing and promoting practical embedded systems research at the Department of Computer Science. It was executed during the study period 2012-2013. Aim of the project was to build a showcase robot: a system that allows open design in future robotics projects. As an architecture design goal, it was set that the new platform should be made as simple as possible, yet containing many interfaces based on which external stakeholders could build applications in future sub-projects. The project team planned to also invite Computer Science research teams to exhibit their expertise in building applications for the platform. Initial application ideas included e.g. gesture- and speech recognition, natural language generation and sensor-based ubiquitous subsystems.

The project started in August 12 and continued until May 2013. The core project team spent autumn 2012 investigating hard- and software technologies related to the topic, building prototypes of e.g. speech recognition, movement on-demand and experimental capacitative sensors. Physical construction process of custom circuit boards and -chassis parts was also studied.

Action Plan

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 project’s goals was clear, plans for actualizing the steps for proceeding had 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. 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

After having recognized that the "Revenue Streams"-parameter that had been used in cases A and B was not applicable for the technology-driven project, it was changed into "Requirement Priorities". Despite its commercial connotations, "Competitive advantage" -parameter was left to the canvas to allow contemplation on alternative implementation options. Research hypothesis C.1 was set as: "The Product Vision Canvas and selected Product Vision Framework parameters can be used to rapidly capture vague software ideas".

Analysis of actions

The canvas was introduced to the team at a project meeting in January 2013. During the meeting, discussion was yet again, mainly dominated by solution space details such as project’s outcomes and features. The first principles of physical design were identified:

*The platform should be free-standing and it should not move. It should contain a web camera to provide the system possibilities to experiment with.*

After noticing that the overall product design started to be concretized, author started to sketch the sub-project ideas with the InnoCanvas. During the sketching process, questions about the details of the projects were queried by the facilitator from the project team.
After the meeting, ten subproject sketches were reviewed with the project group and missing information was discussed. Three major aggregate categories of sub-projects 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.

Some days after the group meeting, the project manager was invited separately to review the sketches. He was explained the idea of the InnCa -framework as an analysis tool for creating a clearer description of a development project’s goals and outcome. First, the overall Robotics Lab -umbrella project idea was walked through the design process by iterating the original canvas sketch. As a result, two top priority sub-projects were identified to be "Establishing the Baseline Architecture" (C.P.a) and "Physical Prototype Construction" (C.P.b).

The project manager was asked to choose either of the emergent sub-projects: C.P.a or C.P.b to be elaborated. He expressed that the project participants had a more advanced understanding of the outcome of the baseline architecture plan (C.P.a) and chose the physical construction of the prototype (C.P.b) as his next specification task. He was left alone to work with the InnoCanvas -tool. The project manager outlined the project C.P.b in 20 minutes, after which the Product Vision was reviewed with the Author.

After capturing the sub-projects, author conducted a quantitative survey on perceived uncertainty by the project manager before and after applying the InnoCanvas with selected parameters. The PM indicated that uncertainty about the project’s goals had somewhat decreased. Appendix H describes uncertainty perceived by the PM before and after facilitation.

**Conclusions**

Being very experimental, the embedded systems’ project started with high uncertainty about the project goals and means to achieve them. Therefore the author evaluated it belonging into the "Experimental Research" -category in the Pearson’s uncertainty map [Pea91]. The product discovery -process had been very slow and had required an immense amount of learning before concrete actions started to
First platform specifications started to emerge in January 2013. As the first concrete specifications were emerged, the InnoCanvas-tool was successfully used in identifying sub-projects. Evidence supporting the hypothesis C.1 was found, the project manager had experienced a decrease of uncertainty about the project’s goals and indicated that the tool was applicable for identifying sub-project priorities, leading to comprehension of next steps of the big project. The analysis had also helped to outline the scope of the each sub-project.

A new discovery during the research was that only by looking at the 10 sub-project idea canvas papers, it was obvious that goals of some projects were defined with more precision than others. Some of the sub-project canvases were filled with very detailed and rich information, while others were very scarce. Therefore the hypotheses B.2: "The framework can be used to evaluate the amount of uncertainty about the project’s outcome during a software development effort" received more support.

Example canvases about the project goals in sub-project’s C.P.a and C.P.b are introduced in Appendix E.
5 Discussion

In this chapter, results from research is presented, their implications and limitations are discussed.

Chapter 5.1 presents answers to research questions posed in chapter 1.1.
Chapter 5.2 explains implications of the results.
Chapter 5.3 discusses limitations of results.
Chapter 5.3 drafts directions for future work.

5.1 Analysis

Answering three research questions were taken as a goal of this thesis.

RQ1: What is known about the concept of Product Vision in NPD and SE literature?

A "Product Vision" describes the overarching goal of a software development effort - it’s reason for existence. Performing holistic project front-end activities, i.e. "product envisioning" before starting the development aids to create a shared mental model of the project’s goals and constraints. All assumptions made in the product envisioning process should be validated with project- and end-product stakeholders, such as business owners or end-users. Sufficient project front-end work contributes positively to both project performance and end-product acceptance.

Negligence of software project front-end activities leads to uncertainty about the project’s goals and priorities, effecting the project performance negatively. Overdone front-end work includes excessive planning, which stagnates the development effort unnecessarily. The line in between barely sufficient and overdone software product envisioning is extremely hard to identify. Therefore, iterative and incremental or even concurrent design of a software Product Vision is often required.

Descriptions of benefits, problems and guidelines for product envisioning exist in both practitioner- and scientific body of knowledge. Some prevailing Product Vision frameworks emerge from New Product Development body of knowledge and
some research on integrating them as is in agile software development exists. Existing Product Vision models from Software Development body of knowledge range from simple bullet point descriptions to heuristics for creating highly detailed software designs. A contemporary and popular method for illustrating a Product Vision has established as a single-page canvas poster, containing 6-9 vision parameters.

**RQ2: How to construct a holistic software Product Vision framework?**

As the contribution of this thesis, an integrated Software Product Vision-framework was created. After a thorough literature survey, four established and one experimental Product Vision-model were chosen to be the foundation of this work. Performing a semantic analysis on the Product Vision models revealed significant similarities in their constitution. A detailed comparison of their parameters revealed that all models assess some of the same knowledge, with somewhat differing terminology. Some parameters were identified to be unique for distinct models. Thus, it was concluded that creating a new, unified model for the Software Product Vision would be beneficial.

Parameters from each framework were collected and divided into 23 parameter classes. Later the classes were organized into five integrative meta-categories that helped comprehension of the dimensions of knowledge the new InnCa-framework addressed. Figure 10 displays the resulting Software Product Vision framework. Appendix A displays mappings in between distinct Product Vision-parameters to their classes.

**RQ3: How can the constructed Product Vision framework be applied in real software development efforts?**

Three cases of participatory action research were conducted to evaluate applicability of the framework for software development. In case A, parameters from the InnCa-framework were used as the backbone of a curriculum for designing innovative software concepts with a predefined heuristic. The terminology was found to be understandable for computer scientists and fit for evoking new understanding on whole-product thinking. Selecting parameters relevant to the design task, creating aiding questions and composing them to a simple canvas poster accelerated the soft-
ware innovation process significantly.

In case B, the canvas tool discovered in Case A was attempted to be used to facilitate starting a real software project. However, the author rapidly noticed that the project’s experienced participants preferred their own means of describing the software project’s goals. Instead of active participation, the author decided to use the InnCa-framework parameters to observe reduction of uncertainty about the project’s goals while the project proceeded from project front-end to development phase and later, to the project’s termination.

In case C, the canvas was used to accelerate a very long front-end of innovation process by extracting 10 sub-projects from an experimental embedded systems’ platform development project. Using the canvas poster tool with selected parameters of the framework accelerated proceeding to development from a very long front-end of innovation. Using canvas with the framework helped the project manager to distinguish project priorities and dependencies. It evoked also decisions about concrete first steps for proceeding with the development effort.

Implications of the results are discussed in the next section.

5.2 Implications

The application domain of all research project cases was very different by nature. Each case also was unequal regarding both complexity of the project, uncertainty about goals involved and speed the project was executed.

Case A was an experiment on creating abstract software ideas in a very brief period of time. The Software Innovation Workshop-course was planned to simulate the front-end of a startup-style software project, starting from complete uncertainty about goals of the development effort. The end-user need driven design task was relatively easy and required good person-to-person communication skills, validating assumptions with possible real users of the software product and reflecting concept design decisions accordingly. Even though the workshop resulted in a prioritized list of next to be taken towards the development effort, implementation of the software was never expected. The project consisted of two periods: a 17-day, collaborative
workshop and rapid 7-day individual design task.

Case B involved a real software development effort, the goal of which was to create a working software prototype. A secondary goal of the project was to create a platform that could be used as the basis for a software startup. The case was very serious and executed by high-skilled software developers. Scope of the project was small enough to be achieved during the 3-month project period. The development effort involved minimal planning and predetermined deadline was given for launching the application prototype. The Apps4Finland-competition environment enabled rapid and very concrete acceptance testing of the software product with its real consumer audience.

Case C consisted of embedded systems research for robotics programming experts. The task of designing and building an experimental open design platform was extremely complicated both as a project goal and also as an architecture design task. It required a lot of learning and prototyping before arriving into a conclusion that the development phase of some of the small sub-projects could be commenced. The project was driven strongly by technological opportunities, and it included no external pressure for setting deadlines, scope or acceptance criteria.

5.2.1 Uncertainty About Project Goals in Cases

Each of cases started with a different amount of uncertainty about the project’s goals. In case A, the concept for a new software was yet to be imagined and no preconditions for the design task’s goals was given.

In case B, goals of the project were very clear. Target audience of the mobile open data application was discovered early and the most important use cases and first user interface sketches appeared in the first project meeting. Uncertainty about design decisions stabilized into a "good enough" specification very quickly after the User Experience Designer’s first presentation. After approximately 30 days from starting the development phase, first release of the software’s prototype was made available.
Case C had a crisp, overarching goal from the start. However, during the project front-end phase, new uncertainty sprang from complexity of the project. There was extensive knowledge about technologies and many concrete suggestions for the solution’s features. As the design task was uncomprehendably vast, the progress of defining the project sub-goals and first steps to achieve them was extremely slow. It took over four months to arrive at a phase when concrete steps for building the system could be identified.

In all cases, uncertainty about the project’s goals and means to achieve them was observed to decrease as the projects proceeded. However, the pace was unequal. When teaching whole-product thinking for novices in Case A, the progress stalled and a design canvas -tool was required to help illustrate the path to tangible goals and to outline the product concept. In case B, the adept User Experience Designer did not need help in communicating the product goals and intuitively found a balance in between negligently done front-end analysis and over-specification. In case C, pace of gaining certainty about project about goals was extremely slow due to the complexity of the task, even though expertise of the team was sufficient to implement any of the sub-tasks. After introducing the InnoCanvas with chosen parameters of the framework, progress of sub-project definition became very rapid.

The results imply that in cases A, B and C, the speed uncertainty about a project’s goals was relational to the project’s complexity and expertise of the developers. Voicing distinct aspects of the product concept with the help of the InnoCanvas helped in cases A and C the developers to create a shared mental model about the project’s goals, discuss and iterate it. Introduction of a conceptual framework the developers could attach their tacit knowledge to significantly accelerated the innovation process in these cases.

5.2.2 Adjusting Parameters of the InnoCanvas Tool

Primary drivers of cases A, B and C were very different. Software concepts in Case A were designed after a thorough analysis of software’s end-users’ needs. In Case B, the software product was designed wide consumer market acceptance in mind. Project C instead was solely driven by technological opportunities.
The different biases made applying InnoCanvas in Cases A ad C as is, inappropriate. Some of the parameters that were included in the canvas in Case A, had to be adjusted to follow the needs for Case C. "Revenue Streams" was applicable in the end-user value driven Innovation Workshop (A), but not in the non-profit, technology-driven embedded systems’ project (C). Despite the epiphany, "Competitive Advantage" -parameter was left in the InnoCanvas -poster of Case C where expectedly, it yielded very poor analysis results.

Reflecting on the Interactive Model of Innovation (p.10) by Rothwell and Zegweld [RZ85], software projects are not always motivated single drivers, such as end-user needs, market- or societal pull or technological opportunities. The results from cases A and C impied that fixing a predetermined set of parameters into a canvas tool restricts it's compatibility. Adjusting the InnoCanvas parameters to be included in each design process provided more relevant results.

This conclusion is in contrast with principles of the existing Product Vision models, in which the canvas poster is constitutional. All existing models contain fixed parameters. Some of the models pay heavy emphasis on discovering human end-user- needs while others focus on a more business model driven prioritization. Therefore their use in varying kinds of software projects is limited. The existing canvases are not relevant in e.g. designing systems involving non-human stakeholders, even though these kinds of projects would also benefit from performing a multidisciplinary project front-end analysis.

5.2.3 Amount of Active Product Vision Parameters in Product Envisioning

When planning the curriculum of the Software Innovation Workshop (Case A), Author recognized that to avoid big "up-front design" the amount of Product Vision parameters had to be restricted. Similarly, to avoid negligent concept design, a finite set of parameters had to be required to exist in the design process. In case B, the User Experience Designer executed a software development brief that was generous, but not too restricting for the development effort. Therefore, it was concluded that quantifying an "Agile Optimum" of project front-end activities could prove an interesting research topic of the future.
Some of the Software Product Envisioning Framework parameters (e.g. Solution Outline, Solution Process Analysis) were present from the very beginning of all case studies. This implies that a small set of InnCa-framework parameters may comprise a "Core Product Vision". Referring to implication 5.2.1, a part of the set of parameters should be allowed to be adjusted flexibly. From all parameters, "Constraints" and "Quality goals" were the ones that were least addressed in all cases. Eliciting constraints and quality goals may either benefit from more thorough guidance or their absence may imply that traditional non-functional requirements can be discovered only later during the requirements engineering process.

5.2.4 Value of Generalization in InnCa-Framework’s Parameters

A InnCa-framework parameter "Requirement Priorities" was created by identifying that the following existing framework imperatives produced software feature prioritization criteria:

- What key activities are most expensive? (BMC)
- Problem: Top 3 problems (LC),
- Product: Top 3-5 features (P.PVB)

When applying framework parameter "Requirement Priorities", Case A yielded end-user value driven prioritization. In Case C, the same parameter evoked thorough analysis on sub-project dependencies. This may provide an implication that the generalization of terms that was produced in the semantic analysis described in Appendix 1 provided richer information than the original parameters in these cases.

On Design Heuristics

Three of the established product envisioning models contain a design heuristic that can be used to aid design. A heuristic is a simple set of rules that are often applied linearly to simplify an abstract design process. Main advantages of a predefined heuristic is that they often are impartial in nature and allow experimentation and iteration during the envisioning process. However, in all Cases (A, B, C) the design process was highly iterative and non-linear. A predefined heuristic worked reasonably well for guiding the product discovery novices at Innovation Workshop course
(Case A.) With experienced developers (case B), the design process was led intuitively by an experienced User Experience Designer and thus could not be directed. In case C, the framework was used to rapidly capture sub-project ideas. When observing workflow on both capturing the concepts with the whole development team and elaborating them with the project manager, it was noticed that the design process was again, rather random and iterative than linear. As a conclusion, inventing a predefined heuristic for applying InnCa is in all types of projects is not encouraged and therefore out of the scope of this thesis.

**Comprehension of the terms**

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 and defining the information that was required to establish their meaning. In all cases, the discovery of the following parameters led to the ability to describe a compelling value proposition for the software product:

- Problem Outline
- Solution Outline
- Stakeholder Discovery
- Stakeholder Value Analysis

Despite its special vocabulary, using the framework in the context in case in teaching innovative software project front-end activities to computer scientists was thus, very appropriate.

### 5.3 Limitations of results

This research contains problems regarding both internal- and external validity. Analysis of the established Product Vision frameworks was based on semantical analysis performed by a single individual, it may be subject to bias. Therefore the resulting framework should be peer reviewed and further validated.

Interpretation on whether a single framework parameter appeared in e.g. Case B
was highly subjective, as the canvas tool was not used consciously in capturing it. Meanwhile, in other cases, it was not straightforward to draw the line in between Problem and Solution Process Analysis. Therefore, further specification of each parameter should be considered.

Action research conducted in this thesis is highly relational to the distinct characteristics of each project and actions of the Author described in cases A, B and C. Therefore, the research cannot be repeated with the same results. Also, the scarce amount of cases limits coverage of conclusions of this thesis only to the cases described in Chapter 4.

5.4 Future Work

In her possible PhD studies, Author aims to create new in-depth understanding of the next generation, iterative and incremental software requirements engineering.

5.4.1 Shared mental models about software project’s goals.

Research theme A focuses on creating new understanding about contemporary software Requirements Engineering. The work develops the InnCa-framework further and evaluates its validity in industry cases. An analysis tool for measuring and visualizing uncertainty about a software project’s goals is created and validated.

RQ1: How to create a holistic, Product Vision-based framework for iterative and incremental software requirements engineering?

The framework created in author’s MSc thesis (InnCa1) is reflected in current industry practices and existing software tools. Special attention is paid on identifying goals for new software projects by discovering knowledge required in each company for ending the pre-planning phase and starting the software development effort (go/no-go decisions). Case research with six industry collaborators results in a full article (A1) about the amended InnCa-framework (InnCa2) and its relevance.
RQ2: How to integrate the InnCa2-framework in software engineering education?

The results from RQ1 are used in facilitating software development teams to discover goals of the development effort collaboratively with the software project customer in capstone projects (Ohjelmistotuotantoprojekti). The research aims in evaluating applicability of the InnCa2-framework into software engineering education. Stakeholder collaboration is facilitated with a simple canvas-based tool, helping to track the software project’s goal. The research results in a full article (A2) about teaching pre-project activities as a part of software engineering education.

A secondary goal for RQ2 is to lay grounds for understanding quantification and measurement of software project’s goals, preparing the work for RQ3.

RQ3: How can the amount of uncertainty about a software project’s goals be quantified?

Building on research questions RQ1 and RQ2, controlled experiments in Software Factory on integrating the InnCa2-framework into a real agile software development process is assessed. The research results in full article (A3) about quantifying amount of uncertainty about a software project’s goals with the enhanced Product Vision (InnCa2)-framework in a very simple software project organization.

Systematic gathering of project data on decisions about sources and substance of project goals and software architecture design is initiated to lay grounds for RQ4.

RQ4: How to create a tool for gathering data and assessing amount of uncertainty about project goals in software development? What kind of visualizations can be yielded?
Reflecting the work of RQ3 to existing Information Systems and industry practices for knowledge management of project-, risk- and decision making data of software project goals. Outlining a simple, web-based tool to help a small and agile software development organization to identify, trace and maintain the most important product- and architecture design decisions. Study results in a research paper (A4) about the instruments’ applicability for risk- and uncertainty management in a very small project organization in an educational setting.

RQ5: What kind of tools and visualizations of project goal uncertainty are relevant to industry?

The tool created in RQ4 is validated in 3-6 real cases and reported in a full article about the visualization tool and it’s relevance to industry (A5). As a research method, participatory Action/Design research is used to facilitate learning the tool and get input on it’s relevance.

RQ6: How to maintain gathered information after development is started? Research aims to give tools how to maintain gathered information after the software development phase has been started after the initial product. This is evaluated together with research of RQ5.

5.4.2 Modern Software Requirements Engineering as a complex system of collaborative learning

RQ7: How to create an initial complex system model of collaborative learning about software project goals?

Building on research questions 1-6, a simple model for learning about the project goals is built to create understanding of the sources for Developer Experience (DevX) in a software development process. Validity of the -model is assessed in three industry cases. Research results in a description of the initial CAS as a research paper (A5).
RQ8: Is the CAS-model of requirements engineering relevant to industry?

Evaluating the relevance of the CAS model is performed in three industry cases. Results are presented in a full article, building on the results from RQ7.

6 Conclusions

Uncertainty about a software project’s goals and priorities complicates the development effort. Decisions made in a software project are effected by the software project’s environment. Volatility of requirements may also be effected by varying stakeholder goals or changes in the market- or technology spaces. Intra-project factors such as project management or competences of the product owner and the software team also effect the development effort. Uncertainty is often reflected to lack of knowledge about the project’s goals. It thus complicates making decisions about the project’s priorities. Uncertainty can be reduced by creating and sharing new knowledge [NT96].

While the emphasis of Agile methods are focused on a single development team working on a single project [KL08] and close collaboration of the development team and it’s stakeholders [Amb02], they currently lack support for initial and long-term product planning. 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 [KR06].

Researchers and practitioners have awaken to recognize high-level product envisioning and -planning as an amendment to the Agile development process. Some Product Vision models exist in New Product Development body of knowledge [Moo91, CE09, OP13, Mau12]. An Agile Product Management practitioner expert has published his initial, Product Vision model [Pic13].

In this thesis, the existing Product Vision models were analyzed to find both similarities and differences. It was discovered that all methods approach a great deal of
the same questions with different terminology. Therefore 23 parameters were chosen to represent the knowledge each existing Product Vision model addresses. Based on the analysis, a new Product Vision framework (InnCa) was created, enclosing all parameters that were common and all that were unique to some of the models. Five integrative meta-categories: Problem Space, Stakeholders, Solution Space, Business Space and Project Space were created to make the new, integrated Product Vision Framework comprehensible. The framework was accompanied with a canvas-based analysis tool (InnoCanvas) to help concretize goals and facilitate the initial software product design during development efforts.

The new framework was evaluated in a university course of innovative product discovery for software engineers, as well as in two software development projects for Master’s level students. The results from Case A imply that the Software Engineering curriculum can be complemented with teaching pre-project activities with the help of the InnCa-product envisioning framework. In case B, the framework was applicable for evaluating the amount of uncertainty about the project’s goals in a software project. In case C, the framework was successfully used to capture vague ideas and thus reduce uncertainty about a project’s goals.

Explicit leashing of uncertainty in software projects is not a goal that should be aimed at with research. Software innovation relies on benefiting from volatility: reacting to weak signals and non-rational risk-taking. Instead, we should strive for creating new understanding about uncertainty and prepare students of Computer Science for facing the countless opportunities for software innovation and finding an intuitive "Agile optimum" for requirements engineering activities in new software project initiation.
References


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
<th>Publisher/Year</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
<th>Authors</th>
<th>Publisher</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pic13</td>
<td>All things product owner</td>
<td>Pichler, R.</td>
<td>URL <a href="http://www.romanpichler.com/blog/">http://www.romanpichler.com/blog/</a></td>
<td></td>
</tr>
</tbody>
</table>


Appendix A. Mapping of terms in between the proposed and existing Product Vision statements

<table>
<thead>
<tr>
<th>Problem Space</th>
<th>Existing Product Vision Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>InnCa</td>
<td>Problem Outline</td>
</tr>
<tr>
<td></td>
<td>InnCa</td>
</tr>
<tr>
<td>Problem Outline</td>
<td>Problem to solve (BMC, P.PVB), Problem (LC), The &lt;product name&gt; is a &lt;product category&gt; (PV)</td>
</tr>
<tr>
<td>Problem Process</td>
<td>Problem (LC), Journeys, Epics (P.PC)</td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Existing Product Vision Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>InnCa</td>
<td>Stakeholder Discovery</td>
</tr>
<tr>
<td>Stakeholder Discovery</td>
<td>For &lt;target customer&gt; (PV), Target market (IPD), Customer segments, Key Partners (BMC) Customer segments (LC) Target group (P.PVB)</td>
</tr>
<tr>
<td>Stakeholder Goal Discovery</td>
<td>Who &lt;statement of need or opportunity&gt; (PV), Goal to achieve (P.PVB), Personas (P.PC)</td>
</tr>
<tr>
<td>Stakeholder Value analysis</td>
<td>Key benefits to be deliveres (PV), Customer realtionships (BMC), Business value (LC)</td>
</tr>
<tr>
<td>Stakeholder Prioritization</td>
<td>Most important customers (BMC)</td>
</tr>
<tr>
<td><strong>Solution Space</strong></td>
<td><strong>Existing Product Vision Models</strong></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>InnCa</td>
<td>Product concept (IPD), Product <em>&lt;name&gt;</em> is a <em>&lt;product category&gt;</em> (PV), Key activities (BMC), Solution (LC), Product (P.PVB)</td>
</tr>
<tr>
<td>Solution Outline</td>
<td>Features, High-level specs (IPD), Key activities (BMC), Solution (LC), Journeys, Epics (P.PC)</td>
</tr>
<tr>
<td>Solution Process</td>
<td>Value Proposition (IPD, BMC, LC), Compelling reason to buy (PV), Vision statement (P.PVB)</td>
</tr>
<tr>
<td>Analysis</td>
<td>Attributes, requirements (IPD), Key metrics (LC), Constraints (P.PVB)</td>
</tr>
<tr>
<td>Value Proposition</td>
<td>Attributes, requirements (IPD), Key metrics (LC), Constraints (P.PVB)</td>
</tr>
<tr>
<td>Constraints</td>
<td>What key activities are most expensive? (BMC), Problem: Top 3 problems (LC), Product: Top 3-5 features (P.PVB)</td>
</tr>
<tr>
<td>Quality Goals</td>
<td>Channels (BMC, LC)</td>
</tr>
<tr>
<td>Requirement Priorities</td>
<td>Design (P.PC)</td>
</tr>
<tr>
<td>Delivery channels</td>
<td>The <em>&lt;product name&gt;</em> (IPD)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Market Space</strong></th>
<th><strong>Existing Product Vision Models</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>InnCa</td>
<td>Unlike <em>&lt;primary competitive alternative&gt;</em> (PV)</td>
</tr>
<tr>
<td>Competitor Discovery</td>
<td>Positioning (e.g. price target) (IPV), Our product <em>&lt;statement of primary differentiation&gt;</em> (PV), Unfair Competitive Advantage (LC)</td>
</tr>
<tr>
<td>Competitive Advantage</td>
<td>Cost Structure (BMC, LC)</td>
</tr>
<tr>
<td>Cost Structure</td>
<td>Revenue streams (BMC,LC)</td>
</tr>
<tr>
<td>Revenue Streams</td>
<td>Key metrics (LC), Value: business goals (P.PBV)</td>
</tr>
<tr>
<td>Business Metrics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Project Space</strong></th>
<th><strong>Existing Product Vision Models</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>InnCa</td>
<td>Ready stories (P.PVB), Top3 features (LC)</td>
</tr>
<tr>
<td>First Steps</td>
<td>Project scope (IPD), &quot;Lean canvas is a pathway to MVP&quot; (LC)</td>
</tr>
<tr>
<td>Scope</td>
<td>Key partners, Key resources (BMC)</td>
</tr>
</tbody>
</table>
Appendix B. Aiding questions for Applying the Software Product Vision Framework Parameters

<table>
<thead>
<tr>
<th>InnCa-parameter</th>
<th>Aiding question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Outline</td>
<td>Identify a problem to be solved.</td>
</tr>
<tr>
<td>Problem Process Analysis</td>
<td>How does the problem manifest itself?</td>
</tr>
<tr>
<td>Stakeholder Discovery</td>
<td>Who experience the problem? Who can help to solve it? Who are the key partners?</td>
</tr>
<tr>
<td>Stakeholder Goal Discovery</td>
<td>What are top goals or needs of the stakeholders?</td>
</tr>
<tr>
<td>Stakeholder Value Analysis</td>
<td>How does the proposed solution fulfill needs of the stakeholders?</td>
</tr>
<tr>
<td>Stakeholder Prioritization</td>
<td>Who are the most important stakeholders? [OP13]</td>
</tr>
<tr>
<td>Solution Outline Description</td>
<td>Describe the solution at a high level. Summarize it to a few sentences.</td>
</tr>
<tr>
<td>Solution Process Analysis</td>
<td>How does the solution work?</td>
</tr>
<tr>
<td>Value Proposition</td>
<td>Summarize the key benefit of using the system in a brief sentence.</td>
</tr>
<tr>
<td>Requirement Priorities</td>
<td>Name one feature the system can’t survive without?</td>
</tr>
<tr>
<td>Delivery Channels</td>
<td>How do the stakeholders reach the solution?</td>
</tr>
<tr>
<td>Constraints</td>
<td>What limitations do we have to take into account?</td>
</tr>
<tr>
<td></td>
<td>What factors may restrict the project from proceeding?</td>
</tr>
<tr>
<td>Quality Goals</td>
<td>What properties does the software have to meet in order to fulfill the stakeholders’ goals? What non-functional characteristics should the software fulfill in the future?</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>Detailed design is not encouraged in Product Vision.</td>
</tr>
<tr>
<td>Product Name</td>
<td>How would you name the software product?</td>
</tr>
<tr>
<td>Parameter</td>
<td>Aiding question</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Competitor Discovery</td>
<td>What are the competing solutions for solving the same problem?</td>
</tr>
<tr>
<td>Competitive Advantage</td>
<td>How does your solution differentiate itself from it’s competitors?</td>
</tr>
<tr>
<td>Revenue Streams</td>
<td>Identify strongest benefits for each stakeholder. Find revenue streams.</td>
</tr>
<tr>
<td>Cost Structure</td>
<td>What expenses</td>
</tr>
<tr>
<td>Performance Metrics</td>
<td></td>
</tr>
<tr>
<td>Project Resources</td>
<td>What existing resources can we utilize to fulfill the value proposition?</td>
</tr>
<tr>
<td>First Steps</td>
<td>What are the first concrete actions we need to do in order to start the project?</td>
</tr>
<tr>
<td>Scope (project)</td>
<td>What is an initial product description? Which features or properties are included in the next releases?</td>
</tr>
</tbody>
</table>
Appendix C. Process description from the Innovation Workshop (Case A)

Figure x: The Software Innovation Workshop Process
Appendix D. Two example product concepts from the Innovation Workshop (Case A)

Helppo valtakunnallinen liikkuminen

Julkaisu 9.5.2012, julkaisutu Juha Louhiranta

Ongelman kuvaus: Julkaisessa liikenteessä on monia toimijia. Usean eri liikennöijän alueella tapahtuva matkustaminen ja matkustaja liikennöijien välillä tehtävät saamattomat. Järjestelmän tehtävä on mahdollistaa mahdollisimman helppo ja nopea matkustaminen.


Sidosryhmät

Normaalikuluttaja Jantteri:

- Haluaa löytää reitin matkallensa ja maksaa sen.
- Haluaa tarkastella vahvempia vaihtoehtoja (halvin, nopein jne.).
- Haluaa tarkastella lipputietoja.
- Haluaa seurata matkan edistymistä

Liikennöijät:

- Jokaisella on oma rajapinta aikatauluille, osalla on oma maksujärjestelmä, osalla ei.

Täärämmät kilpailijat:

- Vastaavia palveluita, joista voi hakea reitin ja maksaa sen, ei toistaiseksi ole omassa maakunnassa.

Yksittäisten sidosryhmien saamien hyödyt:

- Makupalvelu: oma siivu liikenteestä, lisää volymiä liiketoimintaan
- Pienenmäki liikennöijät, jotka eivät ole osa muista liikenteen järjestelmiä
- Yksittäiset liikenteen palvelut

Kilpailutsu:


Product backlog

1. Tavis haluaa löytää nein reitin matkallaan ja maksaa sen.
2. Tavis haluaa tarkastella vaihtoehtoja (nähän, nopaan jne.).
3. Tavis haluaa tarkastella lipputietoja.
4. Tavis haluaa seurata matkan edistymistä
Laatutyökalut kaikille kohtuuhintaan

Julkaisu 22.5.2012, julkaissut Anni O VahtorantaOngelman kuvaus


Ratkaisuehdotus


Sidosryhmät ja hyödyt:

ASUKAS

● saa käyttöönsä kattavan valikoiman laadukkaita työkaluja.
● saa uudelleen käyttöönsä tarpeettomaksi käynytä työkaluita vapautuvaa varastoa.
● saa työkalujen käyttöön palveluntarjoajan pikaoppaan talvella ja kunnossa järjestelmän toimesta.

TYÖKALUTUKU:

● lisää myyntiosuuttaan halpistyökaluliikkeiden kustannuksella.

TALOYHTIÖ:

● väärittää asukkaiden toholtaan johtuvista ongelmista järjestelmän kautta helposti hoituvan konsultaation avulla, nostaa asuntojen arvoa siinä vaiheessa, kun järjestelmän brändöysä on ylikuultaneen

VAKUUTUSYHTIÖ / TYÖKALUJEN HUOLTAJA:

● lisää asiakkaita järjestelmää varten räätälöitydiltä vaikutuksellistaa

Julkaisun katsomus ja esimerkkikäyttötapaukset

Tulovirrat

Asukkaat maksavat esim. yhtiövastikkeen yhteydessä kuukausimaksua, joka on riippuvainen työkaluvarastoon laajenneen ja muista sovituista palveluista. Ruuvioksiin laskutus voi toimia esim. puhelinlastakin yhteydessä, kuten joissakin juoma-automaateissa, tai käyttöä varten olevasta kooltaasta.

Toteutusvaihtoehtoja ( jotka voivat toimia erinomaisena )

● Talonmies/työkaluvastaava tarjottaa työkaluvarastoa: Talonmies syöttää lainatiedot manuaalisesti järjestelmään ja huolehtii työkalujen

● Itsepalveluvirastot: Työkonni yhteisissä tiloissa kiinteä erikoisvalmistettu työkaluminen. Asukas syöttää lainatiedot päätelaitteella (varaston kiinteä päätte / makoautaalin), jolloin työkalun turnistavan pidikkeen lukitus avautuu ja asukas saa työkalun käyttöönsä.
Appendix E. Two example canvases from Case C
Appendix F. Reduction of uncertainty in case A (Software Innovation Workshop) as a subjective evaluation by the Author.

Beginning = At the beginning of the workshop
End = After the workshop
Appendix G. Perceived uncertainty by the software team in case B (Open Data Application Development) before and after the development phase.

Beginning = At the beginning of the development effort
End = After terminating the project
Appendix H. Perceived uncertainty by the Project Manager in case C (the Robotics Lab) before and after the facilitation

Beginning of project = At the beginning of the mid-project facilitation
End = After specification of sub-projects with the InnoCanvas.
Appendix I. Questionnaires for evaluating perceived uncertainty in Case B

A bird's-eye view on your software project.

Hello, welcome and thank you. By answering this survey, you contribute to the empirical part of Hanna's thesis. Please try to answer all questions and leave a comment if you don't understand the question. The word “product” refers to the end-product of your recent Innopaja software development effort.

Name of the product;

What was your role in the project?

Problem
You helped to create a software. Which end-user need or problem does it address?

Solution
How does the software solve this problem?

Stakeholders
Who are the stakeholders? Name all people and systems involved.

Competitive advantage
What makes the software product unique?

Value
How would you summarize the product into one sentence?

Critical process
What is the most important use case or process in your software? A one the product couldn’t exist without.

Prioritization criteria
What are the priorities for further development of the software? (They can be e.g. business constraints, quality requirements, legislative restraints, end-user demands?)

Next steps
Name three properties or features that should be developed next?

Other comments
Epävarmuustutkimus - Taukkis

* Required

**Projektin nimi** *

Oma rooli
- Dav
- UK

Kun projektin aikoi, sen lopputulos oli minulle
- Täysin epäselvä
- Kohtuullisen selvä
- Täysin selvä

Kun projektin aikoi, sen tekninen toteuttamistapa oli minulle
- Täysin epäselvä
- Kohtuullisen selvä
- Täysin selvä

Kun projektin loppui, sen lopputulos oli minulle
- Täysin epäselvä
- Kohtuullisen selvä
- Täysin selvä

Kun projektin loppui, sen taktinen toteuttamistapa oli minulle
- Täysin epäselvä
- Kohtuullisen selvä
- Täysin selvä

Submit