Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

Mikko Parkkola

PRODUCT MANAGEMENT AND PRODUCT OWNER ROLE IN LARGE-SCALE AGILE SOFTWARE DEVELOPMENT

Thesis submitted for examination for the degree of Master of Science in Technology

Espoo 1.3.2010

Thesis supervisor:

Prof. D.Sc.(Tech) Tomi Männistö

Thesis instructors:

Ph.D. Janne Järvinen
Lic.Sc.(Tech) Kristian Rautiainen

Aalto-yliopisto
Teknillinen korkeakoulu

diplomityön
tiivistelmä

Tekijä: Mikko Parkkola

Työn nimi: Tuotehallinta ja tuoteomistajan rooli laajan mittakaavan ketterässä ohjelmistokehityksessä

Päivämäärä: 1.3.2010 Kieli: Englanti Sivumäärä: 8+75

Tiedekunta: Informaatio- ja luonnontieteiden tiedekunta

Professuuri: Ohjelmistotuotanto Koodi: T-76

Valvoja: Prof. TkT Tomi Männistö

Ohjaajat: FT Janne Järvinen, TkL Kristian Rautiainen

Tämä tutkielma keskittyy tuoteomistajuuteen, erityisesti tuotehallintoon ja Scrumin tuoteomistajan rooliin, kaupallisessa suuren mittakaavan ohjelmistotuotekehitysympäristössä, jossa hyödynnetään ketteriä ja lean-ohjemistokehitysmenetelmiä ja -periaatteita. Tutkimuksen tavoitteena on ymmärtää tuotepäällikön ja tuoteomistajan rooli, minkälainen yhteys näiden välillä on, miten näitä rooleja voidaan skaalata suurissa organisaatioissa, ja miten ne on toteutettu todellisuudessa esimerkkiyrityksessä.

Teoreettisessa osiossa käydään tiiviisti läpi olennaiset konseptit tuotehallinnosta, ketteristä menetelmistä ja lean-ohjelmistokehityksestä. Tätä seuraa katsaus nykyisestä keskustelusta tuoteomistajan roolista ja Scrumin skaalaamisesta usean tiimin ympäristöihin. Valittuihin näkökulmiin sisältyvät Pragmatic Marketingin, Ken Schwaberin, Craig Larmanin ja Bas Vodden sekä Dean Leffingwellin näkemykset. Näiden perusteella tutkielmassa muodostetaan synteesi, joka pyrkii yhdistämään näkemykset sekä löytämään synergiat niiden välillä. Tapaustutkimus kuvaa tuoteomistajuuden iteratiivisen kehityksen F-Secure Oy:ssä ketterien menetelmien käyttöönoton aikana, jonka jälkeen viimeisintä tilannetta verrataan teoreettiseen synteesiin.

Tutkimuksen tulokset osoittevat tuotehallinnon toimialueella toimivien tuotepäälliköiden roolin olevan eri kuin Scrumin tuoteomistajan rooli, vaikkakin ne ovat toisiinsa kytköksissä. Synteesi esittää näiden väliset yhteydet, sekä mahdolliset tavat vastuiden jakamiseen usean henkilön kesken suurissa ympäristöissä. Tapaustutkimus antaa tukea tälle hypoteesille, vaikkakin joitain eroavaisuuksia havaittiin synteesin välillä. Nämä poikkeamat voivat muodostaa mahdollisuuden organisaation parantamiseen.

Avainsanat: Ketterä ohjelmistokehitys, Scrum, Tuotehallinta, Tuotepäällikkö, Tuoteomistajuus, Tuoteomistaja

Aalto University
School of Science and Technology

abstract of the
master's thesis

Author: Mikko Parkkola

Title: Product Management and Product Owner role in Large-Scale Agile Software Development

Date: 1.3.2010 Language: English Number of pages: 8+75

Faculty: Faculty of Information and Natural Sciences

Professorship: Software engineering Code: T-76

Supervisor: Prof. D.Sc.(Tech) Tomi Männistö

Instructors: Ph.D. Janne Järvinen, Lic.Sc.(Tech) Kristian Rautiainen

This study focuses on the product ownership, namely product management and the Scrum product owner role, in large-scale commercial software product development environments where agile and lean principles and practices are applied in engineering. The goals for this research are to gain understanding about the software product manager and product owner roles, how they are connected, how these roles can be scaled in large organizations, and how they are implemented in reality in a case study company.

In the theoretical part, the key concepts of product management, related agile methods and lean software development are summarized. This is followed by an overview of current discussion about the role of product owner and scaling Scrum to environments with multiple development teams. The selected viewpoints come from Pragmatic Marketing, Ken Schwaber, Craig Larman and Bas Vodde, and Dean Leffingwell. Based on these views, the study presents a synthesis that seeks the synergies and combines these into such. The case study describes the iterative evolution of the product ownership domain of F-Secure Corporation during the period of agile adaptation, after which the latest situation is compared with the theoretical synthesis.

The results of the study show that product manager role(s) operating within the product management domain are not the same as the Scrum product owner role(s), although they are connected. The synthesis presents the relationships between them, and identifies possible means to share the responsibilities between several people in large environments. The case study shows support for this hypothesis although some deviations from the synthesis are identified. These deviations can possibly be an opportunity for improvement in the organization.

Keywords: Agile development, Scrum, Lean, Product Management, Product Manager, Product Ownership, Product Owner

Preface

I want to thank Professor Tomi Männistö and both my instructors D.Sc.(Tech.) Janne Järvinen and Lic.Sc.(Tech) Kristian Rautiainen for the excellent guidance throughout the writing process.

Also, I thank Craig Larman and Bas Vodde for sharing the pre-print of the relevant parts of their upcoming book and thoughts, hints and ideas about the subject, as well as Dean Leffingwell and Pragmatic Marketing for the permissions to use their graphics and diagrams in this thesis. In addition, special thanks to Bret Pulkka-Stone for proofreading and providing recommendations on the language of the thesis.

Finally, I want to present my gratitude to my current and former colleagues at work, especially Pekka Usva, Markku Kutvonen, Vasco Duarte, Towo Toivola and Esa Tornikoski, researchers and fellow students from Software Business and Engineering institute of Helsinki University of Technology, my friends, relatives and family, and especially my spouse Erika for the outstanding support and patience which kept me going.

Helsinki, 1.3.2010

Mikko Parkkola

Contents

Abbreviations and Acronyms

Backlog	Prioritized "Todo"-list of work to be done.
CPG	Consumer Packaged Goods – industry where the product management concept was originally created.
CQE	Chief Quality Engineer
F-LEX	Scrum based software development process used in F-Secure.
FPRP	Rational Unified Process®based software development process used in F-Secure.
Lean	Software development methodology created by Mary and Tom Poppendieck based on the Lean product development, which originates from the Toyota production system.
PdM	Product Manager
PdO or PO	"Product Owner – The Person who is responsible for what the Scrum Team Builds and for optimizing the value of it. The Product Owner is responsible for maximizing the value of the product being developed while minimizing the risk."[Sch07]
PDC	Product Council
PjM	Project Manager
PSG	Project Steering Group
Roadmap	A document that describes and/or visualizes planned future product releases, time of the release and high-level intent for the content.
Scrum	Agile software process framework.
Scrum Master	"The person responsible for ensuring that everyone on the Scrum Team follows the Scrum process and rules, and who removes impediments to the success of the Scrum Team."[Sch07]
Sprint	Time-boxed development iteration in scrum.
SM	Solution Manager
SPjM	Solution Project Manager
TPdM	Technical Product Manager
XP	Extreme Programming – collection of Agile engineering practices.

1 Introduction

During the past decade, a development methodology called Agile software development has gained an increasing amount of interest from the software industry. Agile software development methods are becoming more and more popular, as they provide light-weight solutions to many traditionally problematic areas: scheduling, resourcing, requirement management, and risk management to name but a few. Agile methods, especially Scrum, were created from the development team's point of view. Scrum's primary purpose was to protect the team from the changing environment and conflicting priorities for a timeboxed period of efficiently spent working time. Currently the most used agile practices are based on the Scrum process framework, which defines the development process, main roles and responsibilities, and process improvement practices.

Although Scrum is widely adopted and has proven useful, it is not yet mature – while larger organizations start adopting Scrum, the areas which are not yet fully covered in the current framework are unveiled as problems are discovered. Fundamental problems currently being researched are the scalability to an enterprise-size organization and product ownership in this environment.

The product ownership problem is related to the scalability of the framework, and the philosophically different approach when utilizing Scrum in a product development based software industry. Considering the single customer based contractual manufacturing industry which was the primary birthplace of the framework, it is perhaps unsurprising that scalability becomes a problem when the complexity and size of the organization, product portfolio, and the product itself exceeds the general size order generally met in such an environment.

1.1 Problem Statement

Scrum defines three roles: the development team, scrum master and the product owner. While Scrum defines the process and responsibilities very well for the development team and the scrum master role, the product owner is not well covered. Another non-covered area in Scrum is scaling the framework for multiple teams collaborating towards the same product. These two areas are currently being researched and investigated, with the aim of finding suitable practices in order to adopt Scrum to a large scale agile product development environment; the focus of this study is in the product ownership structure in such a relatively large enterprise. The questions where this study tries to find answers are the following:

- What is the role of the product manager?
- What is the role of the product owner?
- Is the product owner the same as product manager?

- How does the scrum product owner scale up, and how the responsibilities can be shared between several persons?
- How a product owner and/or product manager role is implemented in large scale agile product development?

The primary problem is about the definition of the product owner role itself. The product owner, according to Scrum, is the person responsible for the requirements management and the return of investment and is the gatekeeper between the rest of the world and the team.[SB02, Sch09] The product owner in Scrum has a lot of similarities to the on-site customer in XP[Bec99, BA04], who is the customer representative and communication point for the team to give the priorities and to answer questions of any kind for clarifying the desired functionality and characteristics of the product. Such an oracle is very suitable for customer specific manufacturing projects. When doing product development for a wider customer segment, the fact that there is not a single customer but a group or mass of customers with similar needs and yet often conflicting requirements, introduces the challenges of traditional product management to the picture. The product owner is expected to be available for the team and represent all customers and external stakeholders, but on the other hand, the product owner needs to be outbound and spend time with customers and external stakeholders in order to understand their needs in depth. The product owner is a difficult, multi talent role working on multiple levels – the product owner must operate with expertise in both long-term strategic and day-to-day tactical decision making, from the customer facing sales through to the developer facing technical areas. While there is yet no consensus among the industry and current research what exactly the product owner is in product development context, there are similarities in the viewpoints which can be agreed.

The second problem is the scalability of the product owner in an environment where multiple teams are collaborating towards a common goal. As a vividly described BBC case study example by Lowery and Evans shows, it is not trivial and there are multiple opportunities for serious issues. Scrum works very well for a single team, and in small projects where a single team is enough, a single product owner can get the job done. In large projects with multiple teams, a single product owner can not get everything done, as multiple teams multiply the time and effort needed from the product owner towards the development teams. When the workload grows, it is either away from the other stakeholders, or from the teams. In both cases, the end result will be bad, lacking short-term guidance or missing long-term product strategy and vision. If several product owners are introduced to solve the capacity problem, it is not evident how they should be organized to preserve the single point of contact for the stakeholders, including the teams. Without coordination, multiple product owners cannot do it either.[LE07]

1.2 Scope of the Thesis

The focus of this study is to analyze what the agile product owner role means in both a product development and product management context, and how it is applied for one product line at F-Secure Corporation.

The previous research focussed on the known and established theory, with some historical perspective regarding implementation of product management and agile software development. This can be regarded as the basic background information or and introduction to the context of the problem domain and the theories that are applied in the case study environment. The current discussion is a literature study limited to the selected viewpoints of currently active and internationally recognized publishers. Although the viewpoints may be part of a bigger conceptual framework, only the parts which are directly related to product ownership, i.e. product management or the product owner role, are covered.

The case study describes the evolution of product management and ownership in F-Secure corporation. At F-Secure, the organizational structure has evolved through annual transformations. In these reorganizing events, product-related responsibilities have been modified almost every time, providing an interesting story of iterative evolution.

F-Secure has multiple products for different customer segments, and the roles and responsibilities do not always correspond exactly one-to-one in all cases. This study was limited to cover the evolution of the product management from F-Secure's "windows clients" program, and from the perspective of the product line for corporate and business customers. The history of this case study is also limited to personal experience in another related function within F-Secure starting from the year 2005, and starting from 2008 as the product manager for the windows clients aimed at the corporate and business customer segment.

1.3 Structure of the Thesis

The study is divided into seven sections. The sections are organized to provide a reading path from the general description of the context narrowing down to the personal interpretation of the answers to the research questions and critical review of the conclusions.

The first section, *1 Introduction*, gives an overview of the problem area and rationale behind the research questions. The purpose of this section is to explain the context and scope of the research, and provide a reading guide for the rest of the sections for a reader who is interested in specific topics only.

The second section, *2 Research Methods*, describes the methods used in the literature and case studies and how these were validated in respect to the academic research criteria for an MSc thesis.

3 Previous Research is a walk-through of related theoretical background and his-

tory for the context and research problems. This section covers the boundaries and domain area of product management, the commonly used agile methods and lean philosophy. These are then applied in the further research as well as in the environment of the case study.

The *4 Current Discussion* section summarizes several viewpoints and suggested solutions to the research problems. The goal of this part of the study is to delve into relevant current theoretical and empirical research developments surrounding the problem area. Personal analysis and synthesis are presented at the end of this section.

5 Case Study tells the story of how product management has evolved together with the agile transformation in the selected case company. The beginning of the section goes through the history of how the relevant responsibilities were arranged before the start of the larger scale agile adaptation, after which the topics cover several iterations of organizational development year by year until the time of writing is reached. The end of the section explains the situation after the latest changes in more detail.

6 Conclusions wraps up the study and tries to provide answers to the research questions presented in the *1.1 Problem Statement* subsection. The thesis is closed with *7 Discussion* section, that tries to identify the strengths and weaknesses of the study.

2 Research Methods

2.1 Literature Study

The purpose of the theoretical part of this thesis is to collect knowledge of previous work influencing both the current theoretical discussion regarding the research problems and the applied frameworks and values of the environment covered in the case study. Theoretical analysis and synthesis that seeks answers to the four first research questions, i.e. role definitions and the scalability model, is based on the literature study.

Information regarding the basic theoretical background of agile and lean methodologies and scrum was gathered from well known and widely accepted publications. Since the focus problem areas of this study are not well covered in these sources, as described in subsection 1.1 Problem Statement, more recent, but less cited publications were reviewed. As these problem areas are also currently actively researched and discussed beyond the academic publications by researchers and practitioners, selected Internet blogs were also followed and used as sources. Original references and reference paths to cited work within these community-maintained publications were retrieved whenever possible for ensuring authenticity and to gather further pointers for related literature.

2.2 Case Study

The case study was performed according to the recommendations for single case studies suggested by Yin[Yin94]. It attempts to find an answer to the last research question, "How a product owner and/or product manager role is implemented in large scale agile product development?" The case is analyzed against a conceptual analysis formed based on the literature study to evaluate the similarities to the theoretical synthesis. Following this, possible improvement areas that would have importance for the case study company should be visible.

In the case study part of the thesis, information was gathered from the company's internal documents and free form theme interviews with key R&D, business and product management decision makers of the specific subject that were available in the company. Also personal observations were used to describe the areas where personal experiences were directly related to and connected with the topics. All of the case study information, analysis and conclusions were reviewed with the described stakeholders and interviewees to ensure the authenticity and validity of the writing.

The information related to the time of pre-agile structure and dedicated product management function was gathered from personal observations, various interviews and analysis of company internal documents, consisting of job descriptions of the related positions, presentation slides and descriptions of organizational functions.

Information starting from introduction of Product Owner position is collected from personal experience in this role, interviews of and documentation from the available

people from the organization in related positions, with related history knowledge, and key decision makers, such as the head of a business unit and R&D, for the rationale behind the changes.

3 Previous Research

3.1 Product Management

Peter Drucker (1909-2005), one of the most respected management theorists of his time, described marketing in words which could also be used to cover product management:

> *"There will always, one can assume, be need for some selling. But the aim of marketing is to make selling superfluous. The aim of marketing is to know and understand the customer so well that the product or service fits him and sells itself. Ideally, marketing should result in a customer who is ready to buy. All that should be needed then is to make the product or service available. – – Because its purpouse is to create a customer, the business enterprise has two – and only two – basic functions: marketing and innovation. Marketing and innovation produce results; all the rest are 'costs'."[Dru74]*

The concept of product management was started in 1931 by Procter & Gamble, when an executive of the company dedicated a single person to have sole responsibility for the Camay soap product brand, bidding against another more successful Procter & Gamble soap brand, the Ivory soap. The concept was very successful and it was quickly copied by many other consumer packaged goods (CPG) companies.[Gor00, p. 5]

This long history of CPG product management has made it well established within the CPG industry. There is a strong relationship between managing brand(s) and marketing. However, product management principles are adopted also among other industries, with somewhat different viewpoints but sharing many of the same principles. The main interest and focus of this study are in software product management, which is a specialised area of the field of high-technology product management. Compared with CPG product management, high-tech – and software product management especially – has a much stronger relationship and dependency between technology, research and development than its CPG counterparts, as the tailoring of products and services to fit customer needs is a much more engineering-intensive process.

3.1.1 Software Product Management

Ari Tikka from Nokia Siemens Networks summarized his experiences of software product management problems in his presentation at the AgileFinland seminar in 2008. According to Tikka, different functions in the value chain traditionally operate in silos, which are more or less isolated from other functions in the same company. Nevertheless the value is produced by going through this chain all the way from the R&D to the customer. Between these functions, such as R&D, product management,

marketing, sales, and so on, communication usually takes place when a handover happens. These silos generate gaps in knowledge and expectations due to limited collaboration, causing confusion, unclarity, conflicts and lack of transparency.[Tik08]

All the stakeholders in the value chain need to be more connected to make the value delivery successful, but the participants have different cultures and interests, making the collaboration difficult. Tikka highlights three different general organization subcultures that have different conflicting interests, but which all need to be satisfied (Figure 1). These cultures are:

- Business management subculture, with investor interests focusing on market realities
- Customer interface subculture, with customers' interests focusing on market realities
- R&D subculture, with R&D interests focusing on production realities[Tik08]

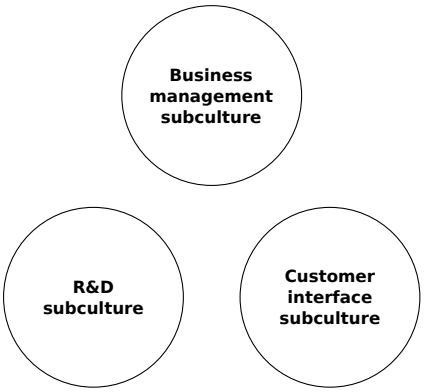


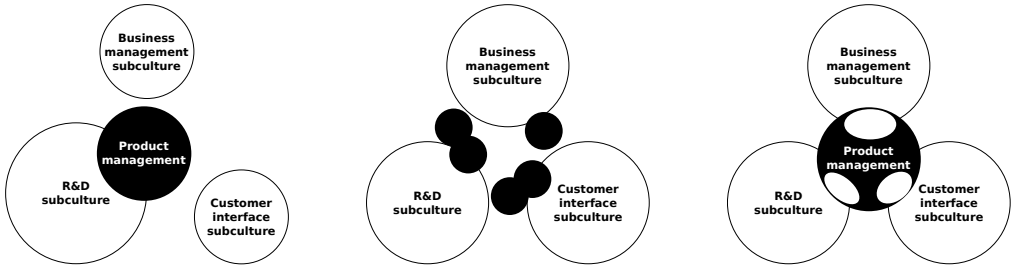
Figure 1: Conflicting organizational cultures, adapted from Ari Tikka.[Tik08]

The viewpoints of these groups are different: in many cases they interpret the importance of the same issue differently.

> *"To stay alive an organization needs to balance between integrity and the three stakeholders' conflicting interests. The subcultures become different, because the members of each culture work daily with different questions. The subcultures may well understand each others rationally, but different things are important. "Yes I understand, but...". This dilemma is present at macro and micro levels, e.g. Company, business unit, department, team... even individual."[Tik08]*

Organizations have tried various ways to solve these conflicts and "fill the gap" between the stakeholders (Figure 2). Giving the power to one of these groups to solve the issues leads to biased dominance (Subfigure 2a). Steering groups typically have representatives from these different subcultures for lobbying their interests

(Subfigure 2b). A separate product management function tries to fill the gap as well, but has a tendency of becoming an isolated unit with the same problems as described above (Subfigure 2c).[Tik08]



(a) Strong function (b) Steering group (c) Separate product management function

Figure 2: Filling the gap between the cultures, adapted from Ari Tikka.[Tik08]

3.1.2 Product Management Domain

For describing and visualizing what is actually included in the product management domain, the framework[Pra09] created by the Pragmatic Marketing training company represents quite well what I have personally experienced while working within this area. The framework identifies product management activities spanning across the strategic–tactical and business–technical dimensions as shown in Figure 3. In the framework, these dimensions are used to fill in the grid of common product management related activities in Figure 4.

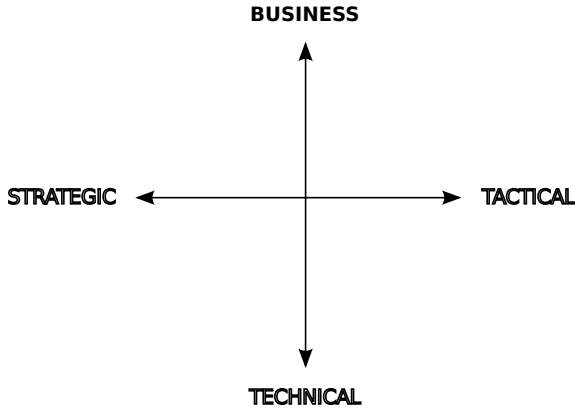


Figure 3: Product management dimensions, adapted from the Pragmatic Marketing Framework^{TM}.[HJM08]

Based on these dimensions Hohmann et al. claim that product manager is typically seen in four different ways by people within the organization:

Technical/Strategic: Research & development teams often consider product management as a technical resource for strategic guidance. The R&D view of

product manager might expect knowledge of market requirements, competitive analysis, technology assessment, user personas and scenarios. In many cases once the teams start to see the product manager as a great source of market information, *"market encyclopedia"* and *"customer expert"*, they want the PdM to be available for them all the time.[HJM08]

Technical/Tactical: Sales utilizes product management heavily as a technical resource for tactical purposes. For sales people, product managers are preferred sources for assisting sales calls, as they can give great product demos, explain the internals of the existing features in customer-oriented language, provide the most recent roadmaps and insights into future plans.[HJM08]

Business/Tactical: Marketing relies on product management to support promotions, product launches and go-to-market plans with tactical, business-oriented information. According to Hohmann et al., in most vendor organizations the marketing department is focused on marketing communications with expertise in promotion, rather than technology. Some technical abilities are needed for correct positioning and messaging, but without the promotional expertise it does not work out either.[HJM08]

Business/Strategic: The executive management team needs product management as a strategic resource for business thinking at the product level, which is seen to be realized through business plans and portfolio & product roadmaps.[HJM08]

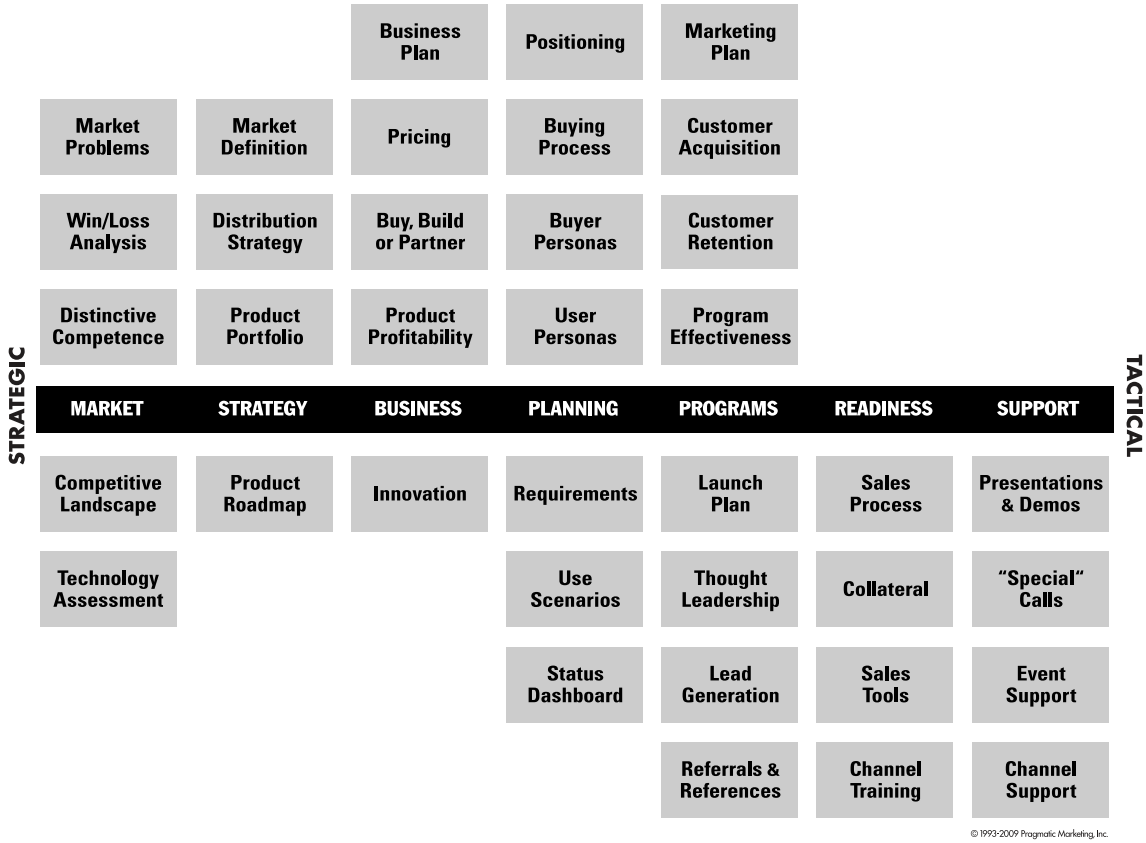


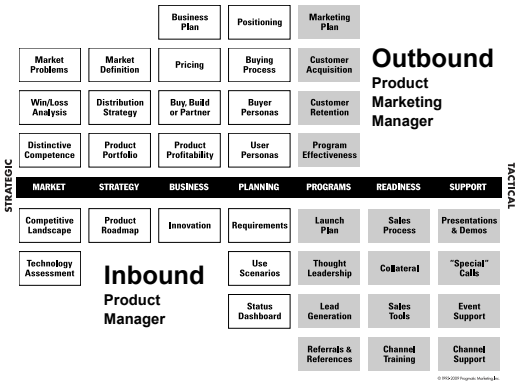
Figure 4: Product management activities, from the Pragmatic Marketing Framework™[Pra09]. Used with permission.

3.1.3 Product Manager – Marketing, Technical, Strategic

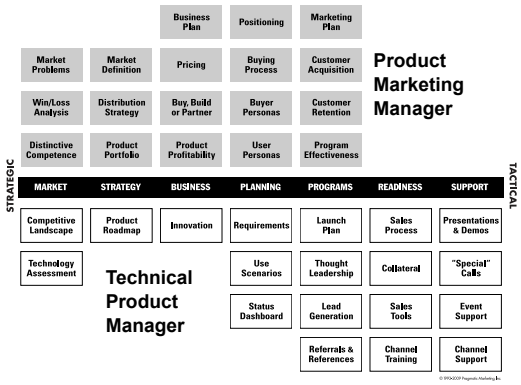
> *"The overall responsibility of a product manager is to integrate the various segments of a business into a strategically focused whole, maximizing the value of a product by coordinating the production of an offering with an understanding of market needs. A product manager must oversee all aspects of a product or service line in order to create and deliver superior customer satisfaction while simultaneously providing long-term value for the company."[Gor00]*

While in smaller organizations where the activities listed in Figure 4 happen in less structured and detailed level, a single product manager might be able or forced to look after the whole product management domain alone. Product managers of this kind are typically referenced as the "CEO of the product", due to the empowerment and significant responsibility attached. However, when the complexity of the product and the size of the organization is high enough, a single person is simply not enough to look into all the areas of the product management domain, forcing a split of the responsibility. This division tends to be into domain areas of specialized product managers working together. For these domain area specialists there is a wide spectrum of different titles that are being used — product manager, product marketing manager, technical product manager, market manager, industry manager, industry strategist, business development, product strategist, product line manager, general manager, and so on — there are no standard ways of mapping the responsibilities to a title or to tell if two persons with the same professional titles are actually doing the same work. Due to these reasons it is important not to focus on the titles used in the product management context but the domain areas.

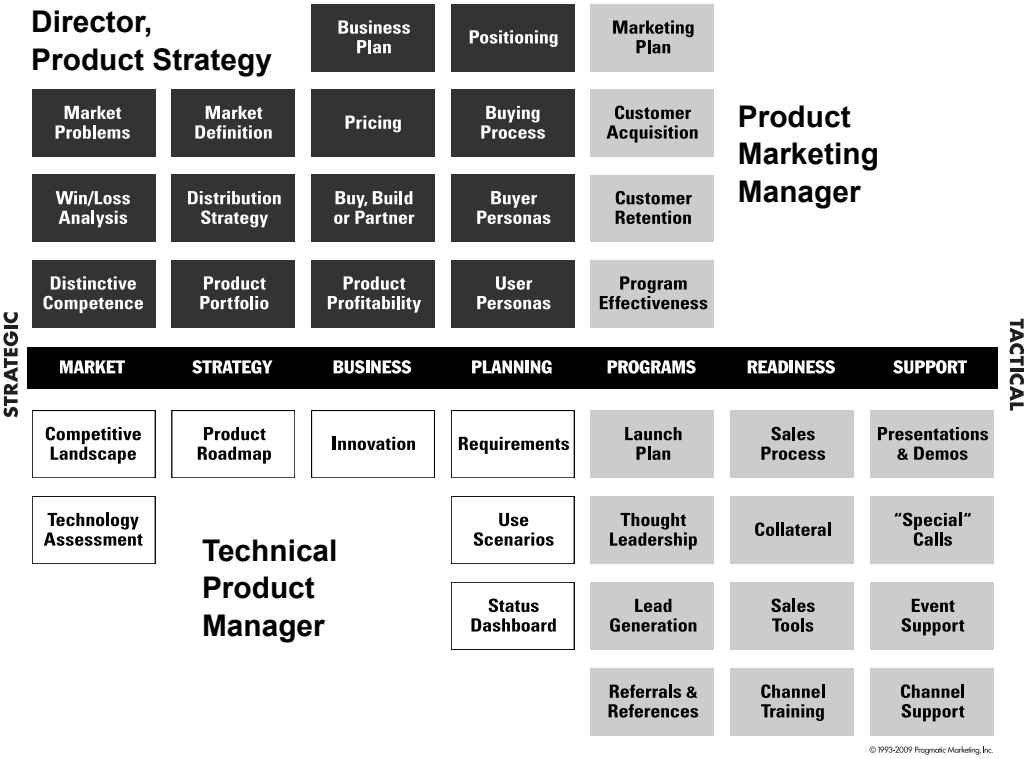
Regardless of the problems with the titles and role naming, there are some more or less common approaches to split the product management domain for two or more persons using the Pragmatic Marketing Framework™, by dividing the responsibility areas based on the dimensional axis as shown in Figure 5. In Subfigure 5a the domain is split between outbound *Product Marketing Manager* role and inbound *Product Manager* role, where the outbound role takes more responsibility for tactical marketing and sales support areas, while the inbound role focuses on the strategic business and technology areas. In Subfigure 5b, the domain is split between the technically oriented *Technical Product Manager* who looks after all the technical issues from strategic to tactical, and business oriented *Product Marketing Manager* who takes care of the business development. Subfigure 5c shows a possible share between three roles, where *Product Strategy Director* owns the business strategy area, and has a team of *Technical Product Manager* for the strategic technology areas, and an outbound *Product Marketing Manager* for the tactical areas. The more complexity and people are involved, the more specialized the focus that the individuals can take.[Joh08b]



(a) Inbound/outbound oriented product manager roles



(b) Technical/marketing oriented product manager roles



(c) Strategic, technical and marketing oriented product manager roles

Figure 5: Splitting the product management domain using the Pragmatic Marketing Framework™[Joh08a, Joh08b, Pra09]. Used with permission.

3.2 Agile Software Development

At the beginning of 2001, 17 representatives of various new software methodologies signed and posted on the Internet a statement referred to as the Agile Manifesto.[Hig01] The manifesto was:

> *"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> ***Individuals and interactions*** *over processes and tools*
> ***Working software*** *over comprehensive documentation*
> ***Customer collaboration*** *over contract negotiation*
> ***Responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more."[BBvB+01]*

In addition to the values, the group also formulated twelve principles as the basis for these new methodologies. For the complete listing of these principles, please refer to Appendix A: Agile Principles.

While the new methodologies were often seen as alternatives to each other, all of them saw the need for light-weight options for existing heavy processes and agreed in principle with the above mentioned values and principles.[FH01].

Nowadays the term "agile methods" is used as an umbrella to cover different methodologies. Interestingly, the methods share common ground, their practices often complementing each other and can be used in many cases together without major adjustments. As a result, the term "agile software development" has established a meaning implying the utilization of one or several of these methods in the software development process in any combination. In a comparative analysis on agile methods Abrahamsson et al. pointed out that out of nine agile methods they investigated, none of them covered all of the product development lifecycle phases fully. The methods were more specialized in providing solutions to certain problem areas or phases, while touching the rest only on an abstract level if at all, and suggested further method specialization and conceptual harmonization[AWSR03]. In another comparative analysis Cohen et al. predict that the agile methods will see some level of consolidation in the future[CLC04].

3.3 Scrum

Scrum is a software development process framework created by Ken Schwaber and Jeff Sutherland. It is a holistic method aimed at providing a self-organizing and flexible process to deal with and adapt to a changing environment and requirements. The word "scrum" originates from rugby, where the players from both teams gather

together to form a special interlinked "scrum formation" when a game restart is necessary. The ball is delivered back into the scrum, from which either team may emerge with possession. Each team is equally powerful and represented by an number of participants, and the scrum itself is both intense and mobile.[Int09].

The rugby analogy was first used by Takeuchi and Nonaka in an article describing an adaptive holistic management method for new product development[TN86], which was referred to as Scrum later by DeGrace and Stahl[DS90]. Influenced also by iterative and incremental software delivery documented by Pittman[Pit93] and Graham[Gra95], Sutherland at Easel Corporation and Schwaber in Advanced Development Methods company applied these methods and developed and formulated the process known as Scrum today[Sch95].

As a light-weight management framework for agile software development projects, Scrum is gathering popularity within software industry[Scr09b] and is supported by numerous agile consultants[Scr09c, Scr09a] worldwide. The main advantages of Scrum are the simplicity, which makes it easy to understand, and abstracting out the actual work content, which makes it easily adoptable to any kind of knowledge intensive work, not only for software development.

3.3.1 Sprint

Scrum uses regular timeboxed events to balance the time used between overhead activities and productive work. The "sprint" is a timeboxed iteration during which the development team works independently and without interference and disturbances. The duration of a sprint is typically 2-4 weeks, but can be agreed to be anything from one week to a few months.

Sprint planning Before a sprint starts, the development team together with scrum master meets with the product owner to set the goals for the sprint, and to select the items from the product backlog to be implemented during the sprint. The development team commits to deliver the selected items by the end of the sprint, and the product owner makes sure the most important items are selected for the sprint.

Sprint execution During the sprint, the goals are locked and the planned items from the sprint planning can not be changed until the sprint ends. When the sprint execution is in progress, the development team is empowered to do whatever it takes to reach the sprint goals. The progress is communicated in short daily stand-up meetings called *daily scrums*, where every team member answers three simple questions: what did he/she do after the previous meeting, what is he/she planning to do before the next meeting, and are there any obstacles blocking his/her work. From the process perspective, the sprint execution is otherwise a black box, and scrum does not take a stand on how the team organizes the work to be done.

Sprint closing At the end of each sprint, the team demonstrates the results of the accomplished work of the sprint to the product owner in a *sprint demo*. Only the work that is done is shown, not work that is still in progress. The sprint backlog is then evaluated in a *sprint review* to go through the work that was done during the iteration. After this, the team analyzes the sprint in a *sprint retrospective*, to collect and select improvement ideas for adapting the process for better performance in the next sprint.

3.3.2 Backlogs

A backlog is a prioritized list where the items with most priority are on top. Scrum uses normally two kinds of backlogs: the *product backlog* containing the prioritized list of requirements and feature requests, and the *sprint backlog* containing the items selected from product backlog to be implemented during the current sprint.

Backlog items Items on the backlogs are product requirements that can be implemented by a development team during one sprint. The backlog items are elaborated to enough detail so that a team can start working on it without requiring extra information and clarification. The items are also estimated both from the implementation effort (cost) and the business value (benefit) point of view, and prioritized on the backlog generally so that the items with the highest benefit/cost ratio are on top.

Tasks When backlog items are selected for implementation during a sprint, the team breaks the selected backlog items into small concrete tasks which can be accomplished in a few hours. A backlog item is considered as done once all the related tasks are completed. These tasks are recorded in the sprint backlog under a corresponding backlog item.

3.3.3 Scrum Roles

Scrum framework identifies only three roles: the development team, the scrum master and the product owner. These three roles share all the management responsibilities in the project[Sch07]. Together they form the scrum team, and the members are called *pigs*, everyone else is a *chicken*[Sch09]. The 'pigs' and 'chickens' are derived from the following story:

> *A chicken and a pig are together when the chicken says, "Let's start a restaurant!" The pig thinks it over and says, "What would we call this restaurant?" The chicken says, "Ham n' Eggs!" The pig says, "No thanks, I'd be committed, but you'd only be involved!"[Sch09]*

Development Team The team is responsible for implementing the product requirements into functionality during timeboxed iterations. The team is cross-functional, self-managing, self-organizing unit independently capable of doing everything required from transforming any requirement item into reality, and take collective responsibility of the success of the project and the iterations.[Sch07]

Scrum Master The scrum master is a special management role in Scrum. The scrum master is the person responsible for looking after the development process itself, ensuring that the values, practices and rules are followed. The scrum master is the facilitator and the driver of the practices, with the primary goal of helping the team to improve the velocity and removing any impediments slowing the team down. Scrum master also represents the management to the team, and the team to the management. According to Schwaber, scrum master role is typically assumed by the team leader, project leader or project manager from the previous organization.[SB02]

The scrum master works with the stakeholders to identify the product owner, and later teaches the product owner how to do their job using the scrum framework. The scrum master can be a member of the development team contributing to the sprint goals by performing the sprint tasks, but Schwaber highlights that this can often lead to priority conflicts between the scrum master responsibilities over removing impediments and development team member responsibilities over performing the tasks. Schwaber stresses that the scrum master should never be the same person as the product owner.[Sch09]

Product Owner The purpose of the product owner is to represent the interest of all the project and product stakeholders, and to ensure that the most valuable things are made first[Sch07]. This person is the one who is officially responsible for the outcome of the project, and is the gatekeeper for the change requests from the stakeholders from the development team point of view, and the single point of contact to the development team from the stakeholders point of view[SB02].

The product owner is the person who "owns" the product backlog. The product owner's key responsibility is to create, manage and maintain the product backlog, and is the only one allowed to prioritize it. The change requests coming from the stakeholders are translated into backlog items, which are then prioritized among the rest of the items on the product backlog by the product owner.[SB02]

> *"For commercial development, the Product Owner may be the product manager. For in-house development efforts, the Product Owner could be the manager of the business function that is being automated. – – The Product Owner can be a Team member, also doing development work. This additional responsibility may cut into the Product Owner's ability to work with stakeholders. However, the Product Owner can never be the ScrumMaster."[Sch09]*

Schwaber highlights that the development team follows only one product backlog, and the product owner managing it is a single person, not a group, to prevent and solve the priority conflict situations which are about to emerge if the team needs to follow multiple backlogs or listen to multiple product owners simultaneously.[SB02]

3.4 Extreme Programming (XP)

Extreme programming is one of the most adopted agile methodologies within the software industry. It was created by Kent Beck, who published the XP method the first time in 1999[Bec99], and later adjusted and improved it in 2004[BA04]. XP is a collection of best practices for software development on a very practical level. As XP does not overlap much with any project management methodology, such as Scrum, it can be easily used together with other methods with more focus on that area. The 12 XP practices are:

1. *The Planning Game – Quickly determine the scope of the next release by combining business priorities and technical estimates. As reality overtakes the plan, update the plan.*
2. *Small releases – Put a simple system into production quickly, then release new versions on a very short cycle.*
3. *Metaphor: Guide all development with a simple shared story of how the whole system works.*
4. *Simple design – The system should be designed as simply as possible at any given moment. Extra complexity is removed as soon as it is discovered.*
5. *Testing – Programmers continually write unit tests, which must run flawlessly for development to continue. Customers write tests demonstrating that features are finished.*
6. *Refactoring – Programmers restructure the system without changing its behavior to remove duplication, improve communication, simplify, or add flexibility.*
7. *Pair programming – All production code is written with two programmers at one machine.*
8. *Collective ownership – Anyone can change any code anywhere in the system at any time.*
9. *Continuous integration – Integrate and build the system many times a day, every time a task is completed.*
10. *40 hour week – Work no more than 40 hours a week as a rule. Never work overtime a second week in a row.*

11. *On-site customer – Include a real, live user on the team, available full-time to answer questions.*

12. *Coding standards – Programmers write all code in accordance with rules emphasizing communication through the code.[BA04, p. 54]*

As in Scrum, XP also operates in incremental deliveries over short iterations and plans are frequently reviewed and updated. In the other overlapping areas, the *On-site customer* is a person representing the user of the system, available for the team for clarifying the use cases and requirements whenever needed. These responsibilities are somewhat similar to as what the product owner has in Scrum when acting as "the voice of the customer".

3.5 Lean software development

Lean software development methodology, or just "Lean" in short, was formulated by Tom and Mary Poppendieck adapting the lean manufacturing values and principles in software development context. Lean philosophy originates from the Japanese manufacturing industry, mainly from the Toyota Production System [PP03]. Lean manufacturing aims for sustainable competitive advantage through the continuous improvement and optimization of the production process for maximum end customer value creation while minimizing the effort and lead time. The philosophy highlights the long-term thinking, optimizing the process through the elimination of waste, developing organization by encouraging the development of individuals, and driving organizational learning by addressing the root causes of the problems[Lik03]. This is realized through following the principles of the Toyota production system:

1. *Base your management decisions on a long-term philosophy, even at the expense of short-term financial goals.*

2. *Create a continuous process flow to bring problems to the surface.*

3. *Use "pull" systems to avoid overproduction.*

4. *Level out the workload (heijunka). (Work like the tortoise, not the hare).*

5. *Build a culture of stopping to fix problems, to get quality right the first time.*

6. *Standardized tasks and processes are the foundation for continuous improvement and employee empowerment.*

7. *Use visual control so no problems are hidden.*

8. *Use only reliable, thoroughly tested technology that serves your people and processes.*

9. *Grow leaders who thoroughly understand the work, live the philosophy, and teach it to others.*

10. *Develop exceptional people and teams who follow your company's philosophy.*
11. *Respect your extended network of partners and suppliers by challenging them and helping them improve.*
12. *Go and see for yourself to thoroughly understand the situation (Genchi Genbutsu).*
13. *Make decisions slowly by consensus, thoroughly considering all options; implement decisions rapidly (nemawashi).*
14. *Become a learning organization through relentless reflection (hansei) and continuous improvement (kaizen).[Lik03]*

3.5.1 Lean Principles for Software Development

The Poppendiecks adapted the original lean principles to seven principles for software development as *eliminate waste*, *build quality in*, *create knowledge*, *defer commitment*, *deliver fast*, *respect people* and *optimize the whole*[PP03, PP06].

Eliminate waste *"Waste is anything that does not add value to a product, value as perceived by the customer."*[PP03] Eliminating waste means actively looking for finding waste in work, and removing that to make the workflow as simple as possible, and focused on having only value adding activities in. The seven wastes in software development according to the Poppendiecks are (corresponding manufacturing wastes in the parenthesis): *partially done work (in-process inventory)*, *extra features (over-production)*, *relearning (extra processing)*, *handoffs (transportation)*, *task switching (motion)*, *delays (waiting)*, *defects (defects)*.[PP06]

Build integrity / quality in In the first book, Poppendiecks describe the concept of integrity as *perceived integrity* and *conceptual integrity*. Percieved integrity being how the product combines the functionality, usability, reliability and economy in a way which delights the user, and achieving this depends on how well the information flow between the development team(s) and the users works. The conceptual integrity is how the different parts and concepts of the system work smoothly together, and achieving this depends on how well the information flow works between the developers and development teams. Maintaining the integrity requires iterative and incremental approach to problem solving and continuous re-factoring.[PP03] In the second book, Poppendiecks converted this to quality, and immediate reaction to quality defects. They claim *"If you routinely find defects in your verification process, your process is defective."*[PP06] The proposed answer to this is *"do it right the first time"*, which means building the process to prevent defects. At Toyota, this was referred to as *stop-the-line*, which means that the root cause for defects is fixed on the spot when they are discovered, to prevent the rapidly accumulating re-work

that results from letting the defects stay in the system and move further in the value stream flow.[PP06]

Create knowledge / amplify learning Software engineering and product development is about creating new knowledge, in contrast to manufacturing where the goal is to reduce variation. During the development new problems are discovered, and solving them requires utilizing and combining previously learned knowledge in a completely new way. Thus, in software engineering and product development, knowledge is an asset which makes a competitive advantage. In order to make it sustainable, organizational culture needs to make experimenting, learning and sharing the created new knowledge a disciplined standard practice to outlearn the competition.[PP06]

Defer commitment / decide as late as possible While deciding early is good when a decision can be later reversed easily, irreversible decisions should be scheduled to the *last responsible moment*, which is the last possible chance to decide before it is too late. By deferring the decision, it is possible to keep multiple options open, and choose the best one based on the latest knowledge and facts when the decision is needed. If an irreversible decision is made earlier than it is needed, the risk of lacking critical information that can result in making a bad decision based on assumptions is high. However, commitment should not be mixed with planning, which is preparing solutions for different scenarios: *"Planning is an important learning exercise, it is critical in developing the right reflexes in an organization, and it is necessary for establishing the high-level architectural design of a complex system. Plans, on the other hand, are overrated."*[PP06]

Deliver as fast as possible Fast delivery is the result when the waste is removed. Poppendiecks describe the value of speed in the following way: *"In development the discovery cycle is critical for learning: Design, implement, feedback, improve. The shorter these cycles are, the more can be learned. Speed assures that customers get what they need now, not what they needed yesterday. It also allows them to delay making up their minds about what they really want until they know more. Compressing the value stream as much as possible is a fundamental lean strategy for eliminating waste."*[PP03]

Respect people / empower the team The people doing the work are the persons who understand the details of the work best. Instead of micromanaging by telling what to do and how, leaders should provide guidance by communicating the intent what needs to be done, and trust the people to figure out the details and decide how to make that happen. Moving the responsibility and decision making to the lowest possible level makes it possible to utilize the knowledge and power of many minds together with the last minute details, which would be impossible to orchestrate by a central authority.[PP03]

Optimize the whole Lean thinking emphases which is known as the systems thinking. The key point in systems thinking is not to focus on optimizing a small part of a system but the whole end-to-end value stream. Local optimization of a part of the decomposed system is a failure mode that on the contrary often has a suboptimal impact on the performance of the whole system when the whole value stream is not considered. Setting goals separately for the decomposed elements of the system often makes the real goal to get lost, and does not give the guidance for making trade-offs between them if conflicts emerge. The key is to take one abstraction layer up and find a high-level goal that will drive the lower level targets in the right direction.[PP06]

3.5.2 Lean Thinking House

A popular representation summarizing the Lean is the "Lean thinking house" shown in Figure 6. This is similar to the earlier representation of the production system used internally at Toyota. In this house, the goal is represented as the roof, that is the competitive advantage through sustainable, fast and superior customer value delivery. The foundation that supports the rest is the support of management for learning and teaching lean values and long-term philosophy. The first main pillar is the respect for people, both internal and external of the organization, valuing real teamwork and mentoring for developing the skills of individuals. The second pillar is continuous improvement for the never ending mission of seeking for the perfection. The 14 principles are the core of the lean, summarizing the building blocks of the pillars and foundation of the lean house. The practices are used for producing more useful information and knowledge to out-learn the competition, and reach the goal.[LV09]

Figure 6: Lean thinking house[Lef09a, 15.9.2009][LV09]. Used with permission.

3.5.3 Lean and Agile

Some researchers claim that Agile methods are an instance of Lean[Lef09a, 15.9.2009, 2.10.2009][Mar09]. Kent Beck, the creator of XP, does not agree even though he considers these methodologies related. Beck expresses his views in a direct comment to Martens's blog posting:

> *"I think that Agile and Lean are strongly related, but that they are two different ideas. Lean aims to achieve efficiency through eliminating waste and respecting people. Agility is a by-product in lean as rapid cycles are required to identify and eliminate waste. Agile software development aims to meet the evolving needs of customers through the early and continuous deployment of valuable software. The values, principles, and practices of the two approaches are different, even though complementary."*[Mar09, Kent Beck's comment 22.6.2009]

The same view is shared by James Coplien, who argued in the ScanAgile 2009 conference that the main purpose of Agile is to react to change, perceived by him as "laid back California guitar playing style coding", while Lean emphases careful

disciplined upfront planning and rapid execution. Coplien also referred to Mary Poppendieck's keynote[Pop09] about the very different cultures and contexts where agile methods and lean originate from, agile from the individualistic United States and lean from teamwork focused Japan.[Cop09]

Agile and lean resonate well together as they share the same goals and similar values, but can also be seen as different paths: while traditional agile methods take a stake on how to build processes and practices from scratch and then improve them over time, lean focuses more on continuous improvement of what you have now – whatever that might be – making it a very suitable starting point for agile transformation for larger companies with established processes. As with other methods and frameworks discussed before, lean thinking can be used together with any agile methods.

3.5.4 Lean and Product Ownership

In Lean software development, Poppendiecks raise the importance of leadership as essential contributor for success, and highlight the difference between leaders and managers as described by John Kotter[Kot99] in Table 1.

Managers	**Leaders**
Cope with Complexity	*Cope with Change*
- Plan and Budget	- Set Direction
- Organize and Staff	- Align People
- Track and Control	- Enable Motivation

Table 1: Managers vs. Leaders by John Kotter[Kot99], table comparison by Tom and Mary Poppendieck[PP06]

Poppendiecks refer to the leader person who is driving the product development as the *product champion*, which corresponds to the Toyota's Chief Engineer. This person has the responsibility and the accountability for the key product decisions and explains the product vision to the team(s).[PP03, PP06]

> *" Product champions at 3M and chief engineers at Toyota have a strong sense of product ownership. – – It might seem that a strong sense of ownership would lead chief engineers and champions to exercise a great deal of control over the development of* their *product, but neither of these leaders has direct authority over the people working on the product. They fully understand that leveraging the talents of a large pool of experts is far more effective than trying to control the work. Thus, they lead the development team instead of trying to manage it. It is because of their dedication and passion that champions and chief engineers excel at inspiring technical teams."[PP06]*

In software context, Lean thinking can be used to analyze the requirements management process by mapping the value stream from the customer need to the availability

of the solution. The first part of this value stream is the requirements management and communication of the need to the development team which is able to create a solution. The second part is the delivery chain through the integration, packaging and distribution of the created solution. For the interests of this study, the requirements value stream is in the primary focus. The goal of the great product ownership is to minimize the lead time, handovers, miscommunication, decision making bureaucracy and other wastes from identifying the customer need to implementation.

> *"I consider lean principles to think about the PO role to be essential. In fact, thats why we recommend what we do. Let me clarify.*
>
> *One of the major problems in product development is that the developers don't understand anymore what and why they are building the product. There are too much layers between the requirement donor and the developers, the overview is lost. Sometimes I say, every layer between requirements donors and developers will transform the requirements a little bit and if there are enough layers... nothing of the original requirement is left. (this is in sync with the "Up the organization" view of layers within an organization).*
>
> *Of course, this is the waste of handoff and loss of knowledge that is caused by that. So therefore, the PO role needs to be as close as possible to real customers and to the team so that the amount of layers are minimized. Then, when a PO would become overloaded, then the solution should *not* add additional layers but instead solve the overload in other ways. Requirement Areas and APO is one of these solutions where there is an additional PO, but we didn't created additional handoff. The separation of PO clarification and prioritization is another way of thinking about the PO role that will reduce the load and NOT add extra layers (instead, remove them even as we propose the team themselves can do a lot of the clarification).*
>
> *Thus, thinking of lean waste.. the value stream from customer to developer and reducing hand-off is an important way of thinking related to the PO role." — Bas Vodde (personal e-mail conversation 25.10.2009)*

In the quote above, Vodde refers to the product ownership scaling model created by him and Craig Larman, which we are going to cover in more detail in the next section. However, his comment provides a good background for the thinking behind this model and can be used for analyzing other models that try to solve the same underlying problems.

4 Current Discussion

Traditional product management principles address product ownership issues from one direction only. Scrum defines the product owner, abstracting everything beyond R&D behind this role, which might in large organizations exceed the responsibilities of a whole department. In the following sections, we will go through different views of the domain, how the picture is seen as changing when using Scrum and other agile methods, and what is current thought about the product owner and product manager roles.

4.1 Pragmatic Marketing view

> *"Product Owner is a new role, created and artificially defined by the Scrum Alliance. Product Owners sit with their development teams full-time, elaborating users' stories, managing sprint-level backlogs, expanding specifications, and interpreting product vision. Most product companies already have staff with similar skills, such As Requirements Analyst or Business Analyst." [Mir08]*

In agile software development, a role of *Product Owner* is used for providing customer understanding, elaborating requirements and making product related day-to-day decisions for R&D teams. According to Hohmann et al.[HJM08] the responsibilities defined by Scrum for this role are a subset of the product management domain, as illustrated with another example of product management domain split in Figure 7.

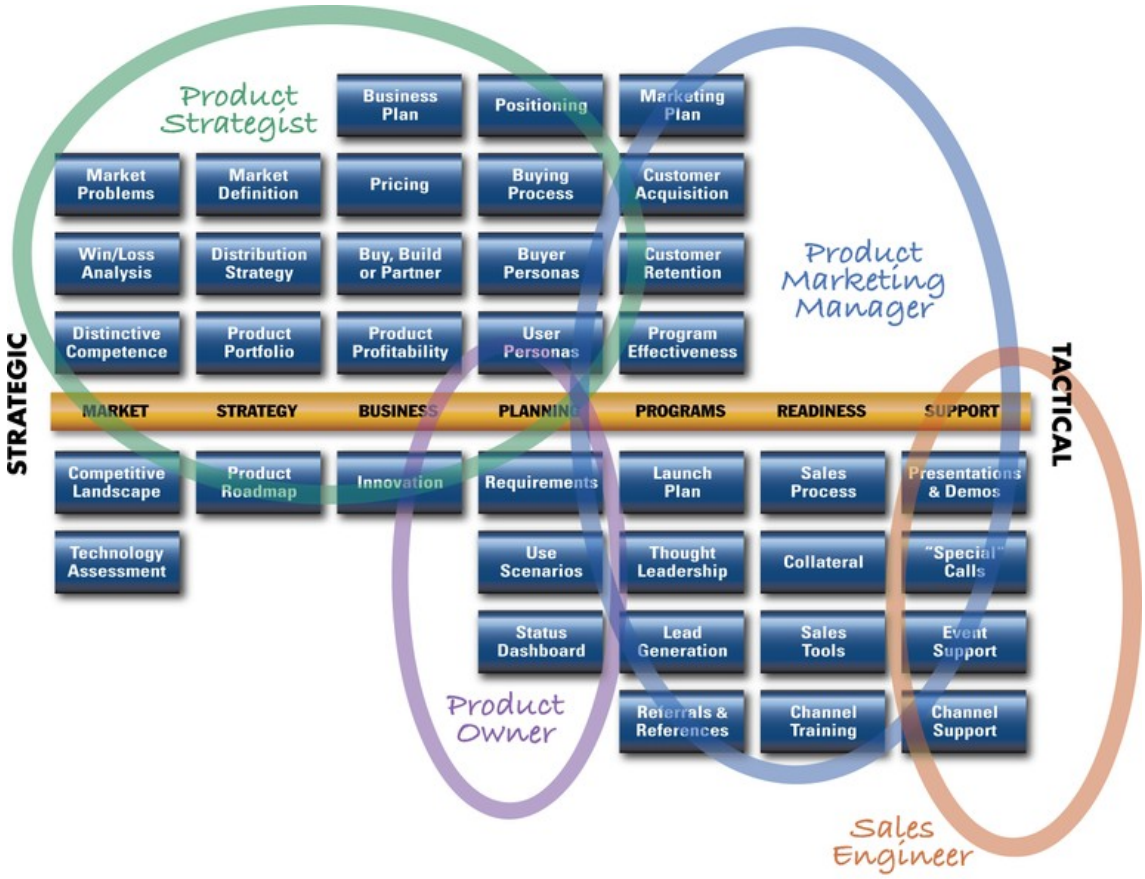


Figure 7: Product owner role in product management using Pragmatic Marketing Framework^{TM}[HJM08]. Used with permission.

Rich Mironov explains the difference between the Product Manager and Product Owner role and their relationship in the following way:

> *"Product Managers are responsible for overall success of their products. This ranges from market strategy to technical tactics to outbound product messaging. For smaller products or smaller companies, one Product Manager covers the entire Framework. This is a challenge, but one that Product Managers take on. With larger products and additional staff, the Product Manager role may be split into technical Product Managers and outbound product marketers, each covering about half of the Framework.*
>
> *Product Managers own any organizational gaps, and cross departmental boundaries to get their products to market. Even when they report up through Engineering, they know they have obligations to customers, Marketing, Sales, and the executive team.*
>
> *Product Owners cover roughly four boxes in the lower middle column: these are mostly technical, semi-strategic tasks, and time-critical to the development team: elaborating user stories, keeping sprints aligned with customer releases, managing sprint-level backlogs, and arranging*

> *customer previews. If a development team has both an Agile Product Manager and a Product Owner, the two must be in tight and constant alignment to avoid whipsawing the team." [Mir08]*

The Pragmatic Marketing view includes the product owner role as part of the product management domain, while some other writers focusing on product management have a more extreme view, considering the agile movement as an internal R&D re-organization which should not affect product management or the product manager's role at all — R&D is seen as a "black box" factory, and what happens in it does not affect product management, even though it is good to know how R&D operates.[Kha08]

4.2 Ken Schwaber's view

Ken Schwaber – one of the originators of the Scrum method[SB02] – did not originally cover the issues of scaling the Scrum framework in environments with multiple teams, but has later published his views on Scrum implementation in an enterprise context[Sch07]. Schwaber seems to continue on simplification, empowerment and self-organization principles to collect and drive the organization in the same direction using simple rules. Similar methods are used in U.S. Marine Corps, where the complexity of the operation is managed by communicating the strategic intent to the teams, empowering them to make required decisions in the chaotic environment of the field of conflict.[PP03, p. 62 –64] Simple rules concerning to control complex systems has an analogy with the swarm intelligence researched by Bonabeau and Meyer[BDT99, BM01], demonstrating how swarms can solve complex problems by individuals following a few basic rules:

> *"Social insects work without supervision. In fact, their teamwork is largely self-organized, and coordination arises from the different interactions among individuals in the colony. Although these interactions might be primitive (one ant merely following the trail left by another, for instance), taken together they result in efficient solutions to difficult problems (such as finding the shortest route to a food source among myriad possible paths). The collective behavior that emerges from a group of social insects has been dubbed 'swarm intelligence"'[BM01]*

The application of simple rules derived from swarm intelligence research has been successfully utilized in solving complex problems – also technical and logistic issues – in a business-critical context, such as routing calls in telecom networks, cargo routing and heating oil delivery. Bonabeau and Meyer also provide an fascinating example where the task of dividing the tasks of scheduling semi-specialized paint booths in the car assembly line was optimized by using the honeybee colony working model. They list flexibility, robustness and self-organization as the advantages of swarm intelligence. A swarm can quickly adapt to changes in the environment, is

operational even if some individuals fail and it does not require heavy top-down control and supervision.[BM01]

Ken Schwaber sees the product owner as responsible for the overall business goals and subgoals, and suggests scaling up the product ownership in multi-team environment by organizing the Scrum Teams in multiple integration levels. Each of these teams then fills all the required roles and knowledge: product owner, scrum master and other team members based on the knowledge required for fully coping with the teams working on more detail and narrow scope level. This concept is illustrated in Figure 8. Schwaber allows re-using some team members of the teams from certain layers as part of the teams on other layers. In this structure, Schwaber suggests that the top level product owner has a Vice President or Director level authority, and is responsible for the overall profit and loss of the project. He or she is assisted by a group of product owners, each of them sharing the same responsibilities on a function level and below. Similarly, a top level scrum master is assisted by a group of scrum masters from the function level teams.[Sch07]

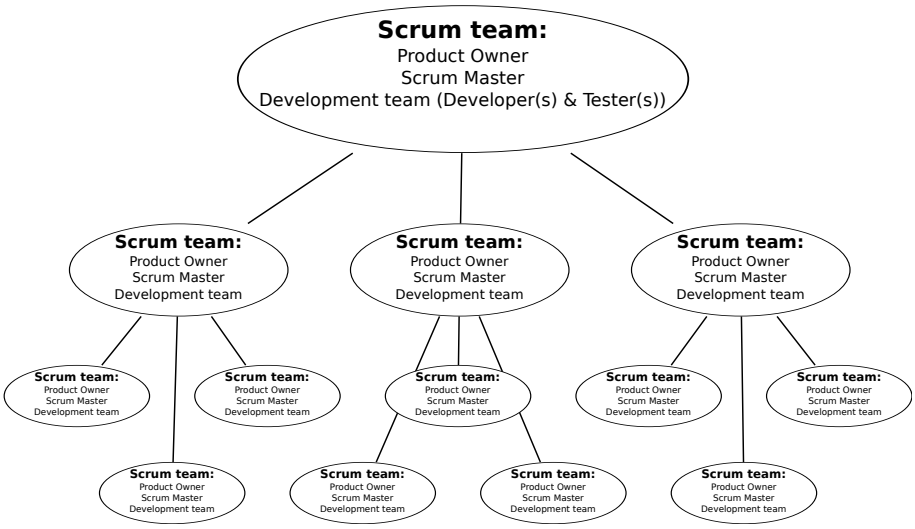


Figure 8: Schwaber's model of scaling Scrum in multi-team environment.

4.3 Craig Larman's and Bas Vodde's view

Craig Larman and Bas Vodde have jointly written a series of publications regarding Agile and Lean implementations in modern software development organizations. Before his consulting career, Vodde was working on agile adaptation at Nokia Networks. Larman has his roots in object oriented development and has written a bestseller containing a managerial overview of the different agile methods[Lar03]. Larman and Leffingwell both try to cover multiple agile methodologies and abstract them out one level higher, to find the root principles behind the similar practices of different methods, and to create a generalized application of the practices. One of Larman's key concepts is the use of feature teams to enable self-organization in multi-team environments. In larger development organizations, highly specialized

component teams are typically arranged around a platform, technology, or architecture, causing a system feature development that affects multiple technical areas go through a series of hand-overs. A feature team is a cross-functional team that has all the required skills and expertise inside the team to carry on a system feature implementation from the beginning to the end with a minimal set of external dependencies. Larman and Vodde use this feature team as the core unit, and scaling up happens by adding multiple feature teams to the picture.

Larman and Vodde strongly suggest having one product owner and one product backlog per product in order to retain visibility and an overview, regardless of the size and complexity of the product. The possible problems that occur from the implementation of this principle are that in larger environments, the product owner is working with too many teams at the same time, the product backlog gets too large to maintain, and the teams start working on the whole complex system. To solve these dilemmas, they suggest grouping the related teams around *requirement areas*. A requirement area is a collection of features that are closely related from the customer's point of view. These requirement areas can be organized to have their own *area backlogs* for related fine grained backlog items which are directly from the product backlog, or subitems for these. The area backlog can be prioritized independently from the other area backlogs by a dedicated *area product owner*. Together with the product owner, area product owners form the *product owner team* that makes product-wide decisions together, while the product owner has the final say.[LV08, p. 217-222]

For large scale Scrum adoptation Larman and Vodde suggest two different kinds of frameworks illustrated in Figure 9 and in Figure 10, depending on the number of teams and the context. When the product owner can no longer manage the overview or efficiently interact with the teams in the first model, they suggest to first try delegating more responsibilities to the teams before deciding to switch to the second model. In the first model, the teams are working with the same, single product backlog. They have common, shared sprint planning sessions with the product owner where the team members can 'bid' the items from the backlog for their teams to the next iteration, and they have shared sprint demos.[LV08, p. 291-297]

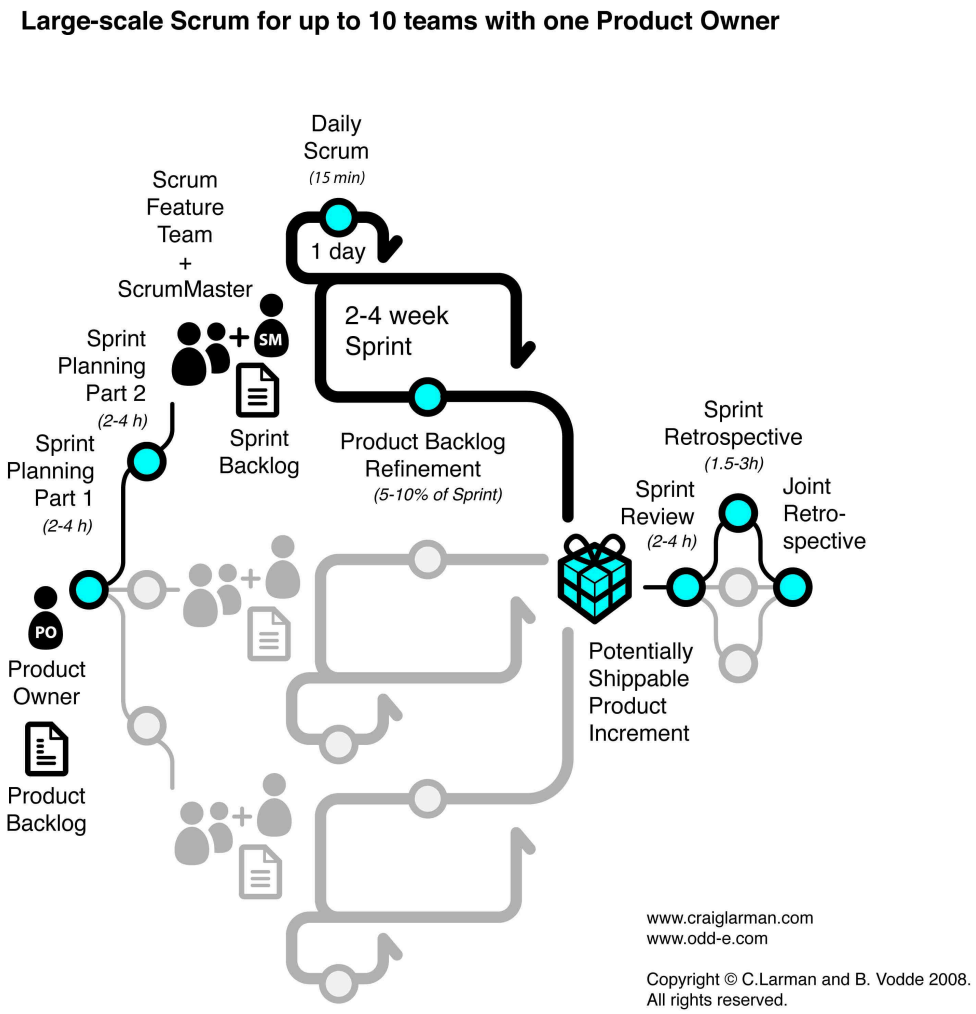


Figure 9: Large-scale Scrum with one product owner by Larman & Vodde.[LV08] Used with permission.

Larman's and Vodde's second model for large scale Scrum multiplies the first model for the requirement areas, each having their own area backlog and area product owner. The area product owners collaborate and the product owner team – led by the product owner – manage the high level product backlog. To properly coordinate the multi-team sprint planning, Larman and Vodde suggest having pre-sprint plannings prior to the iteration for the product owner team to align the objectives for the next sprint, and for the area product owners to discuss priorities across different area backlogs and give and get feedback. For the coordination of teams inside the iterations, Larman and Vodde acknowledge also the Scrum of Scrums practice as a one possible but not mandatory mechanism.[LV08, p. 298-302]

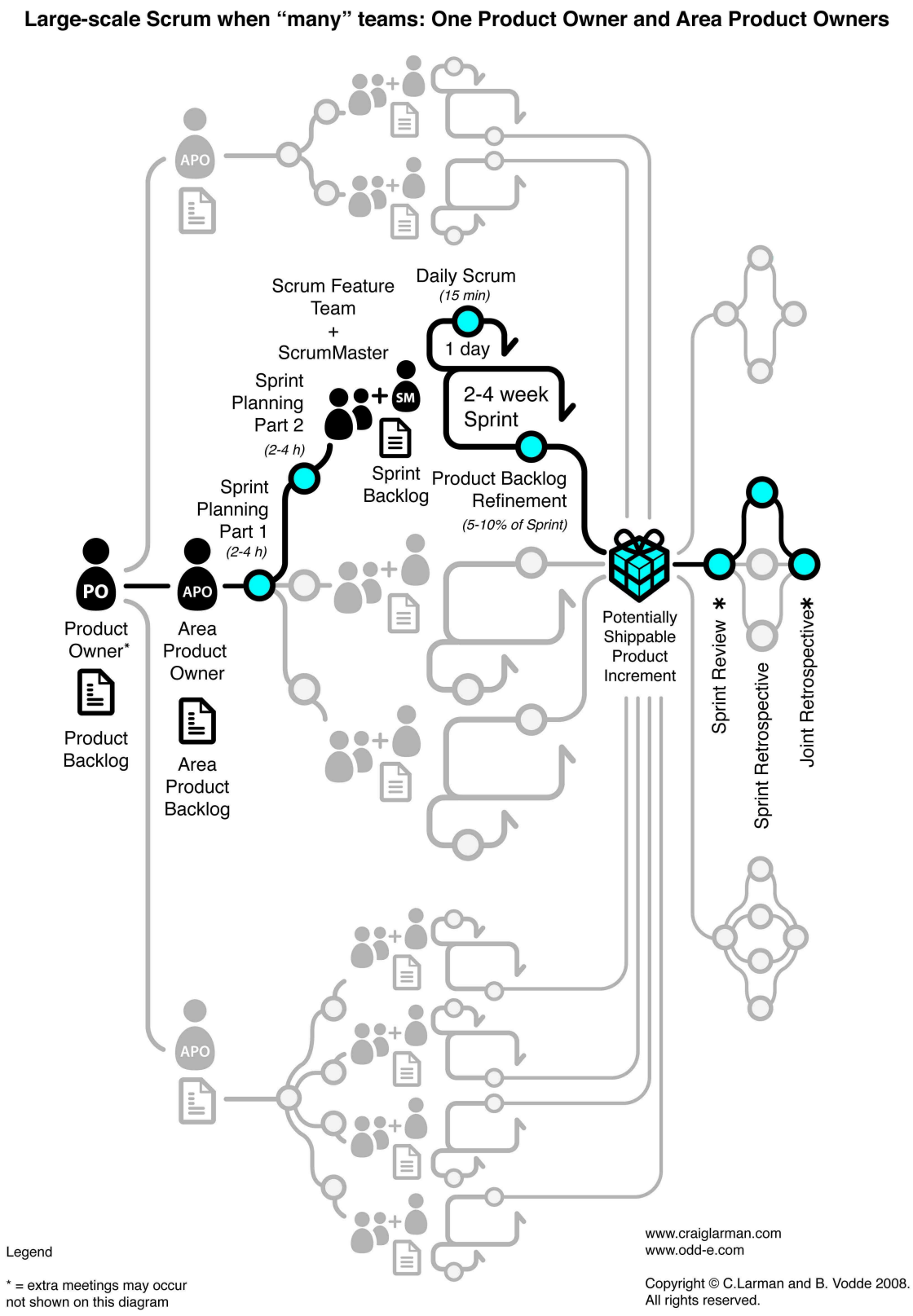


Figure 10: Large-scale Scrum with one product owner and area product owners by Larman & Vodde.[LV08] Used with permission.

4.4 Dean Leffingwell's view

Dean Leffingwell is a senior software engineering consultant with a strong background in enterprise software development processes. Leffingwell has served as the chief methodologist to Rally Software and was the Vice Precident of Rational Software and IBM's Rational Division responsible for the Rational Unified Process (RUP) [Lef09e]. RUP is a commercial framework consisting of best practices and tools for tailored software development processes. RUP has been very popular within enterprise scale software development with tools, certifications and consultation support. Today, Leffingwell supports enterprise scale agile transformations, for example at Symbian and Nokia [Lef09a, 12.7.2008]. Based on this pragmatic research and real-life experience of large-scale software development, Leffingwell has documented his learnings and view of applying agile methods in a enterprise-scale multi-team environment[Lef07, Lef09c, Lef09a]. Leffingwell uses Figure 11 to visualize the ideas of this "Big Picture" in one graph.

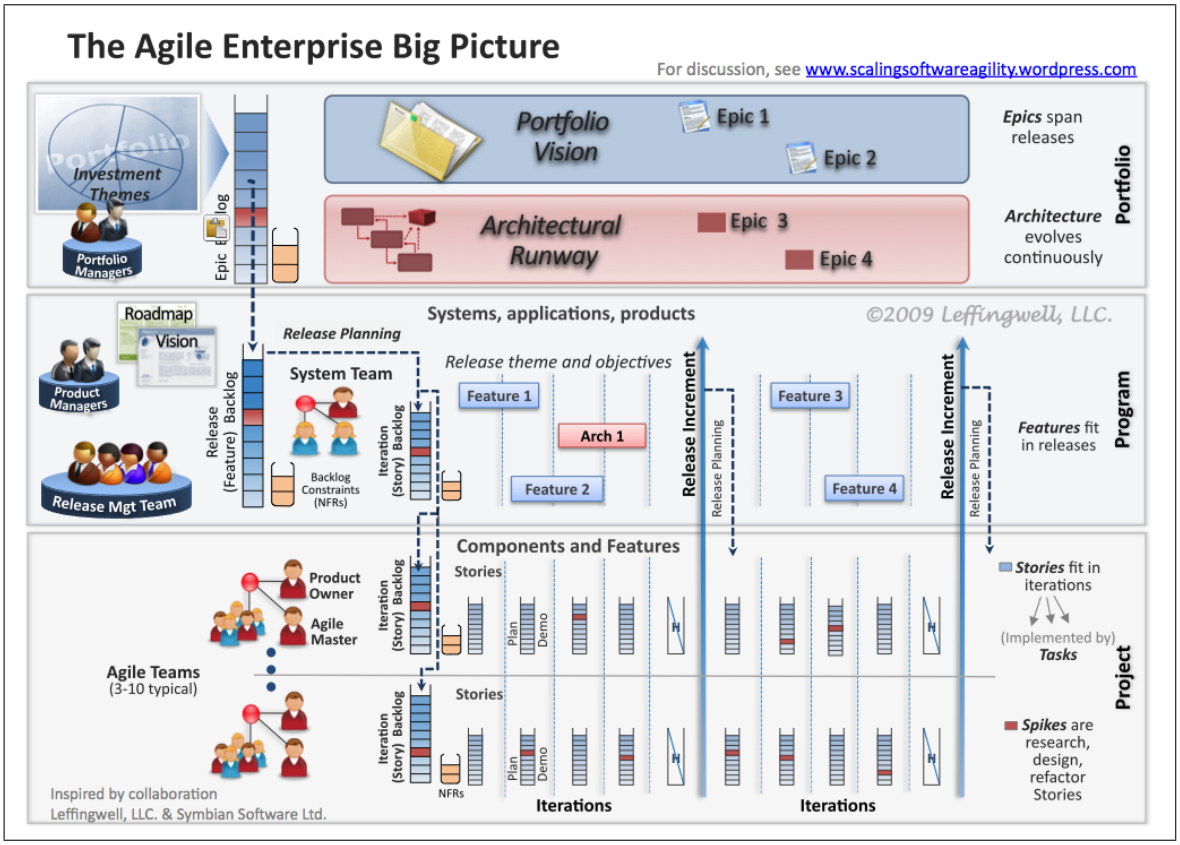


Figure 11: Leffingwell's large scale agile model[Lef09c, Lef09d]. Used with permission.

Leffingwell deliberately separates the Product Owner and the Product Manager roles in multi-team environments. Supporting the Agile Manifesto principle *"Business people and developers must work together daily throughout the project"*[BBvB+01],

Leffingwell criticizes Schwaber's definition of Product Owner by commenting on this quote: *"(Product Owner) is responsible for representing the interests of everyone with a stake in the resulting project ..achieves initial and ongoing funding by creating the initial requirements, return on investment objectives and release plans"*[Sch07] in a following way:

> *"In some smaller organizational contexts, that definition works adequately and one or two product owners are all that are needed to define and prioritize software requirements. However, in the larger software enterprise, the set of responsibilities imbued in the Scrum product owner is more typically a much broader set of responsibilities shared between team and technology-based, product owners and a number of market or program-based,* product or program managers*, who carry their traditional responsibilities of both defining the product and positioning the solution to the marketplace."[Lef09c]*

Leffingwell suggests splitting the product management responsibilities between the market and customer facing agile product manager, and the technology facing agile product owner:

> *"We strongly advocate for Choice #3 [dual agile roles], though it diverges from some current Scrum philosophies. We believe that this puts the right people in the right roles – team-based product owners that work their wonders with the technology – market-based product managers that work their wonders in the market, and it does so with minimum disruption to the enterprise's existing organizational structure. After all, with all we have to change to improve the value stream, why change anything we don't have to?"[Lef09b]*

Leffingwell distinguishes three typical levels where product definition decisions are made: Portfolio, Program and Project levels.

On a *portfolio management* level, *investment levels* are set for different areas, e.g. existing products, new products, future research and managing the end of life of existing products. Investment levels are also set for *strategic product themes*, which represent a set of strategic initiatives affecting portfolio competitiveness. Investment levels differ from priorities, as when following priority order the lowest priority items might never get addressed even if the main goals are successfully achieved, but within the investment mix bandwidth for even the item with the lowest investment is guaranteed. This is important in portfolio management, because lack of focus on areas which require small but continuous investment might not cause issues in short and mid-term, but can ruin the long-term strategic objectives and competitiveness. Besides controlling investment levels, portfolio management defines the long term *portfolio vision* and sets priorities of *epics* within *portfolio backlog*. Epics are initiatives that aim for customer value and they are the highest level of abstraction for the customer needs, and they can be estimated, valued and prioritized separately.

Program management level operates at the release level. Here product management defines *product roadmaps* and set the *product and release visions*, based on the input from the epics and investment levels defined by portfolio management. The product roadmap is a timeline with the product release milestones of one or more related products with realistic goals, e.g. mapping epics for specific releases. Epics are split into *features*, which are more concrete pieces of functionality that realize the related epic. According to Leffingwell, *"The primary content of the (product/release) vision is a set of features"*[Lef09c]. A prioritized *release backlog* of all the planned features is maintained for each release, and is the primary tool for release project tracking for the product managers and for the release management team, a system level coordination team.

Agile teams that are organized around a software component or a feature work on the *project* level. These teams are self-organizing and have cross-functional capabilities for defining, developing, testing and delivering software independently. Each team includes the team-based *Scrum/Agile Master*, who takes the role of management or leadership proxy, and also as the process expert empowers and encourages the team to work towards continuous improvement and greater performance. According to Leffingwell, the *Product Owner* is a team based role as well. The Product Owner and the team elaborate and split the features into smaller user stories, which are testable software requirements that can be implemented by the team within one iteration ('Sprint' in Scrum). The team's Product Owner makes sure the team works on the right things by managing the *iteration backlog* consisting of all the identified user stories and other work the team might be required to work on. Some teams may decompose the user stories into small *tasks* that need to be done in order to complete a story, to help with the effort estimation and tracking within the iteration.[Lef09c]

4.5 Product Ownership Analysis

The product owner, according to Schwaber's original definition, is merely a way to abstract out everything beyond the team boundaries that has anything to do with the definition of requirements and the work contents that the team is about to do. The product owner, by this definition, is a leader within the team that knows what to do next, why, and knows where to find the answer when there is a work content related question that the team cannot solve by themselves. According to Tikka, many Scrum practitioners see narrowly that the product owner:

- *Is the one profit and loss responsible person*
- *Works daily with the team*
- *Is in frequent communication with the customer*
- *Maintains the product backlog*

- *Plus other tasks in a larger product/organization[Tik08]*

Tikka highlights that the product owner is the responsible person for the content, making hard decisions about where to invest time and money:

- *Defines the features of the product*
- *Decides on release date and content*
- *Is responsible for the profitability of the product (ROI)*
- *Prioritizes features according to market value*
- *Can change features and priority between every iteration*
- *Accepts or rejects work results[Tik08, LV10]*

Looking outwards from the team, the product owner is the subject matter expert who is part of the team discussing with the rest of the world outside the team. When looking from outside in, the product manager is a person responsible for the product contents, and other P's (price, place, promotion). When looking at the product management domain, we can agree that the product owner and the product manager can be the same person in environments where the overall product management domain responsibilities are not shared.

In multi-team environments using agile methods, simple mathematics can be used to calculate how many teams a product owner can support if that would be their only responsibility. In theory, around 10% of the sprint is dedicated to sprint planning phases, demos and retrospectives. This corresponds to total of one day out of a two-week sprint. In real-life, therefore, one person could attend the meetings of ten teams if he or she would do nothing else, the meetings were cascaded and do not happen simultaneously. If 10% is added for just-in-time elaboration and backlog prioritization work outside these meetings, the theoretical maximum drops down to 5. This may not be an accurate number, but gives a ballpark of the maximum number of teams a single person can support in theory. When we consider that in commercial product development the product manager can allocate only a portion of his or her time to R&D, this number is reduced further.

Lowery and Evans have documented a case from the BBC where eighty people were organized in nine teams working towards the iPlayer product. The Scrum itself and scrum master role scaled quite naturally by implementing scrum-of-scrums for solving cross-functional impediments and inter-team dependencies, but implementing proper product owner structure was not trivial. They experimented with various product ownership models which did not work: single product owner, two product owners with shared responsibilities, multiple team-based product owners. In each case, the product owners lacked time, knowledge, common vision or coordination.[LE07]

When a single person is not enough to support the development of the product, there is a need for a *product owner team* where the decision-making power is delegated and

distributed. Still, there needs to be a leading person who is responsible for making the final decision when one is escalated – as in military, centralized decision making does not work anymore in modern warfare, and irreversible decisions are delegated to the field.[Tik08]

> *"While having a single Product Owner who has the time and skill to perform all aspects of the position is ideal, real-world dynamics don't always allow for that.*
>
> *In lieu of a full-time PO, esnsuring that there is a single decision maker and distributed coverage of all aspects of the role throughout the broader team — can be a very effective alternative. Point is — don't get stuck in purist thinking. Use common sense and collaboration to get the job done."[Gal09]*

Various researchers and organizations have been scaling Scrum at the same time, and have seen the need for having supporting product owners that assist the overall product owner for a larger number of development teams[LV10]. This exploration has resulted in a confusing use of terminology for these roles, as shown in Table 2.

	Larman & Vodde	various, e.g. Cottmeyer	Schwaber [Sch07]	Pichler, Cohn, Galen	Eckstein	Leffingwell
Overall	Product Owner (PO)	PO	Overall PO	chief PO	lead PO	Product Manager (PdM)
Supporting	PO representative, Area PO	PO proxy	PO	PO	PO	PO

Table 2: Product Owner role terminology mapping. Adapted from Larman & Vodde[LV10], with Leffingwell's terminology added to the original.

It is challenging to find good generic terms to connect the roles discussed earlier. For the convenience of the reader, the following terminology is used in the latter part of this study:

- *Product manager(s)*, and the different specialized product management roles as described by the Pragmatic Marketing Framework™
- *Product owner*, the person who is in direct and frequent communication with R&D during the development of the product and who has overall responsibility over the product requirements.
- *Supporting product owner*, the person having responsibility over a subset of the product requirements, and the person acting as the product owner for a team – managing input to the team's sprint planning and participating in the review and demo – if there are multiple people having a similar role for other teams at the same time.

4.6 Product Ownership Synthesis

This section presents the synthesis over the previously covered viewpoints from a product ownership point of view, focusing on the connecting points and different responsibilities between the different product decision-making layers and directly related roles.

All the analyzed approaches acknowledge the need for splitting product ownership responsibilities, vertically on different levels, and/or horizontally in different areas. The Pragmatic Marketing view has the widest horizontal spread and Dean Leffingwell's view has the biggest vertical depth coverage. In the following synthesis, we will follow Leffingwell's vertical levelling from portfolio management to development teams.

Portfolio level

Dean Leffingwell was only one to address the portfolio management level from the analyzed models. While portfolio management is an important topic, it is a research area of its own that can be excluded from deeper analysis within the product ownership study. Nevertheless, Leffingwell has several interesting points to note.

Portfolio management is a top level authority that makes long-term investment decisions on strategic areas that affect the business performance of the company. In order to do this properly, deep knowledge of the market, environment, technological and financial landscape at a macro level are needed. Implicitly, this responsibility requires high level rank and would resonate well with a vice president, executive team, or a business unit top management level decision making forum, depending on where these decisions are typically made within the company.

The main responsibility of portfolio management is to set the investment levels between business areas, product lines, different products, and strategic portfolio investment *themes*; these are a collection of related strategic initiatives. In an enterprise environment, these decisions can be also made at different levels as appropriate: for example, on an executive team level between business units, at a business unit level between product lines, for product lines between products. The investment level is the granted budget for the specific area, which cannot be moved, borrowed or used for another area. It is the fixed investment into this area for the defined amount of time, in terms of monetary budget, head count or allocation over the planned mid and long-term time interval.

Portfolio management sets the strategic long term objectives for the portfolio, by prioritizing strategic technological and business initiatives. According to Leffingwell's model, these strategic initiatives are called 'epics', but as a generalization, strategic initiative could be also used. These initiatives span across release projects, products and product lines.

Program level

The program level is where Leffingwell placed the product manager's domain. This level operates with reference to individual products, overlooking the time span of several releases into the future. This logically fits the area of the traditional product management domain, which was discussed in subsection 3.1. Pragmatic Marketing Framework™is primarily focussed on mapping the activities and challenges of this level, while other analyzed models leave the details out and abstract them behind the Product Manager role. It is easy to see from subsubsection 3.1.3 that the whole domain of this role is in many cases covered by teamwork. This is the place where the interests of all the stakeholders of the product are combined at a high level.

Hohmann et al. suggest several possible ways to split product management responsibilities among the product management team, but leave the freedom to choose the best possible fit for the context to be considered case by case. The main thrust of their thinking about how to split the domain is to connect closely-related areas under the responsibility of a single role from the map in Figure 4 on page 11, resulting in a group of area specialists that work together as a team. Responsibility for some activities in the borderline of two or several of these roles could and perhaps should be shared, to encourage tight co-operation, communication and thinking outside the box. While there is no clear guidance how this team should practically operate together, there is nothing blocking them from organizing their work using the Scrum framework as well.

Hohmann et al. also map the explicitly defined Scrum product owner responsibilities to the Pragmatic Marketing Framework™in Figure 7 on page 26, which are the planning activities located in the middle of the strategic-tactical axis, on the technical side. These explicitly defined product owner responsibilities reflect the day-to-day interface towards R&D, in the form of backlog maintenance, user story elaboration, and so on. In examples visualized in Figure 5 on page 12, these activities are owned by an inbound product manager or technical product manager. Regardless of the title and the other related responsibilities, the person owning this part of the product management domain is the role that is referenced in agile literature as the product owner, the overall product owner, the chief product owner, the lead product owner, the product manager or the agile product manager. Should there be a technical product manager, it would make sense that this person takes the product owner role during the development project, as shown in Figure 12.

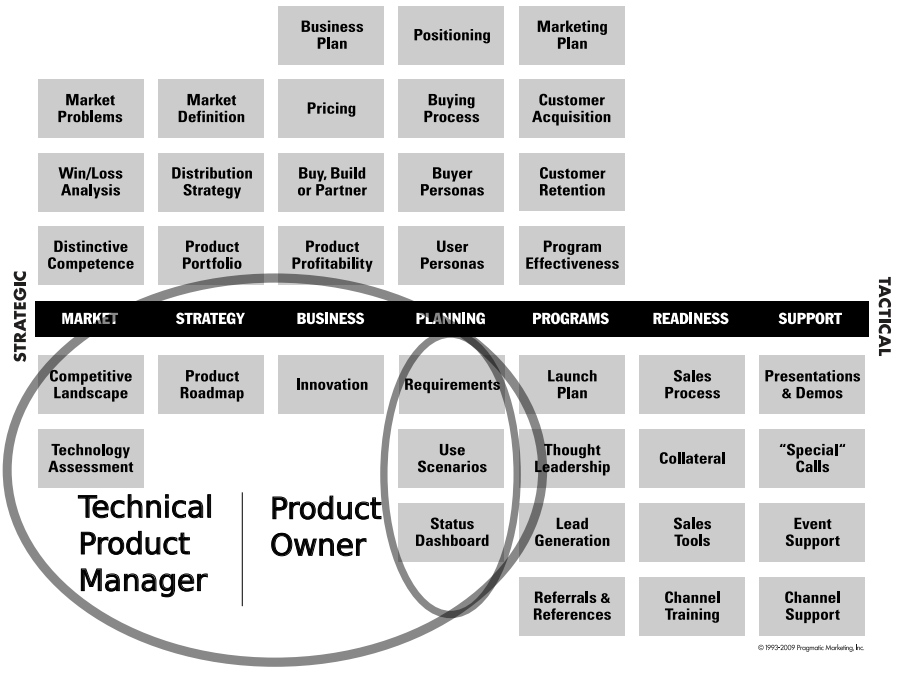


Figure 12: Technical Product Manager taking Product Owner role during development project. Based on the Pragmatic Marketing framework^{TM}[HJM08]. Used with permission.

Project level

Most of the seen agile scaling models concentrate on the project level work. The project level targets towards a product release, over several iterations. A project has a start and pre-defined end criteria. Before the project starts, proper background work for the objectives for the release has been done at a program level and the available resources have been defined by the investment allocation set on the portfolio management level.

The project and the release need to have a responsible owner, the "single wringable neck", who has the overall responsibility over the project outcome and has the authority to make the final decision in escalations. This person is the *product owner*, or the *release owner* for the project. This person operates on both the program and project level. At a program level, this is the person responsible for technical planning activities in the product management domain, as discussed earlier, and who may or may not have other responsibilities as well.

If the number of Scrum teams exceeds the capacity of the product owner, one or more *supporting product owners* are required to serve the needs of the teams. These supporting product owners have a delegated power to make decisions on behalf of the product owner. Naturally, the product owner and supporting product owners need to be aiming towards the same goal, have a shared vision, and co-operatively work together as a team.

Every team needs a product owner or a supporting product owner for content guidance and work acceptance, but a product owner or a supporting product owner can serve efficiently few other teams at the same time. Thus, the supporting product owner can – but does not have to be – a team-based role. It is important that the communication between the team's product owner and the team is free, open and frequent in both directions to build trust and commitment. The product owner or release owner should be part of the product management team at the program level.

It is not clear how the product owner and supporting product owners should share the work amongst themselves. One way of splitting responsibilities might not be applicable to other environments, and this may be dependent on the context. Larman and Vodde suggest splitting the responsibilities based on the customer perceived areas. In this model, each product owner should have enough subject matter expertise and business knowledge to cover the ownership of certain requirements areas, and the teams ideally need to be real cross functional feature teams, having end-to-end delivery responsibility and also being capable of working on multiple levels of the source code. This desirable state might require organizational re-structuring which may not be possible at the given time, thus the current organizational constraints may limit the available options for sharing the work to potentially less optimal, but still functional, combinations.

5 Case Study

5.1 About F-Secure

F-Secure Corporation is a public Finnish software company focusing on data security products. F-Secure is one of the leading vendors for consumer and corporate anti-virus solutions, operating globally and with a strong position especially in the Nordic and European markets.[FHGM09]

Currently the R&D departments of F-Secure employ over 300 people, located in 5 different offices and four countries. F-Secure products support several different operating systems and over 20 different language versions. F-Secure products use common, shared components and product platforms.[PP09]

5.2 2005 – Pre-agile roles and responsibilities

5.2.1 Portfolio level

Product Council The core group of the Product Council was formed of several executive team members and director-level senior management. The Product Council was the top level authority which gave a mandate to the product manager and project manager to drive R&D to realize the approved plan within given boundaries. It also resolved priority conflicts and considered escalations if projects were in danger of slipping seriously in schedule or scope.

The Product Council meeting was organized once a month. In this one full day meeting the core product council members reviewed and approved all the starting product release projects and roadmaps presented by product managers. For the starting projects it was required that:

- the schedule, resourcing and content were planned
- that they followed the standard process and practices
- made sense
- were aligned with the company strategy
- and didn't conflict too badly with other projects.

The Product Council meeting was also a popular knowledge sharing session of all the current states and future plans for product release projects, thus it attracted a lot of other interested audience from various departments throughout the whole company.

5.2.2 Program level

Product Manager F-Secure had Product Managers who were in charge of public facing product communication, as well as making the product related business and technology decisions. During this time, the product manager was understood as "The CEO of the product", being responsible fully for the four P's (product, place, price, promotion). A product manager was typically responsible for one or several products of a product line.[PU09]

Technical Product Manager Technical Product Manager had the ownership and vision for and of several internal components or features, which may or may not be used with several products, owned by one or more Product Managers.

Program Manager The Program Manager was a senior/director level line management position who was also referred to as the resource owner. R&D teams reported to the Program Manager, who was actively involved in the project resourcing, planning, and also driving other initiatives. The Program Manager was and is in a crucial role in change leadership.

5.2.3 Project level

Before the agile transformation, F-Secure was using a modified version of Rational Unified Process®(RUP) as the development process, called FPRP, that was taken into use in 1999[PP09]. RUP was, and still is, a process widely used in professional software development. The process had several connected and overlapping phases, with heavy up-front requirements specification and design phases, following implementation, testing and deployment phases. These phases were not completely sequential, as in the traditional waterfall process, but concurrent, with a smooth focus shift from one phase to the next.

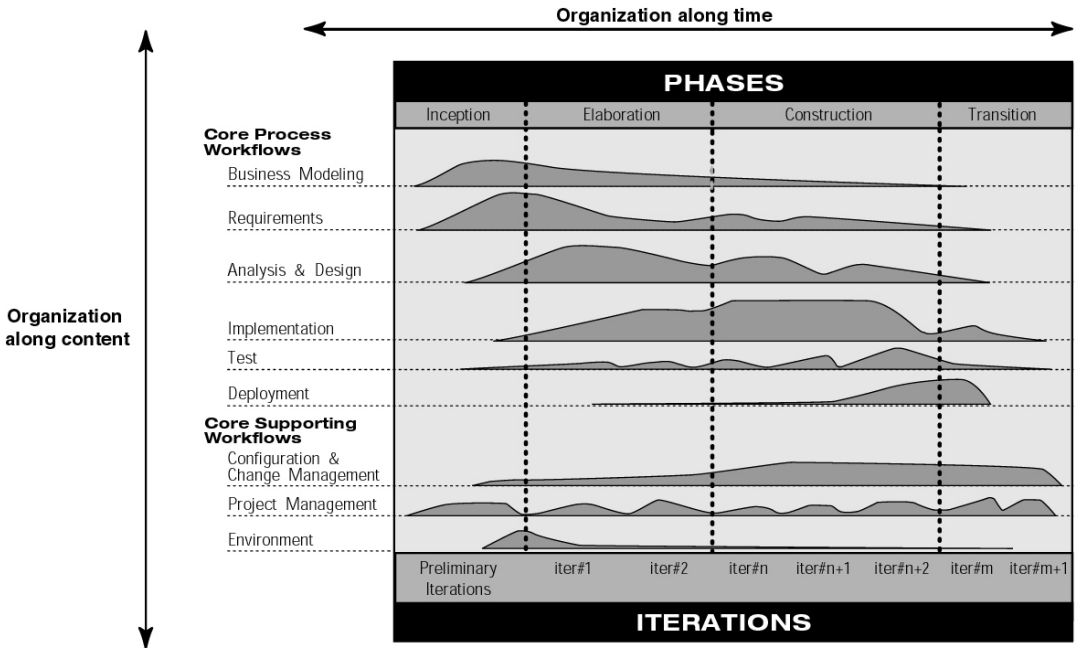


Figure 13: Rational Unified Process®[Rat03]

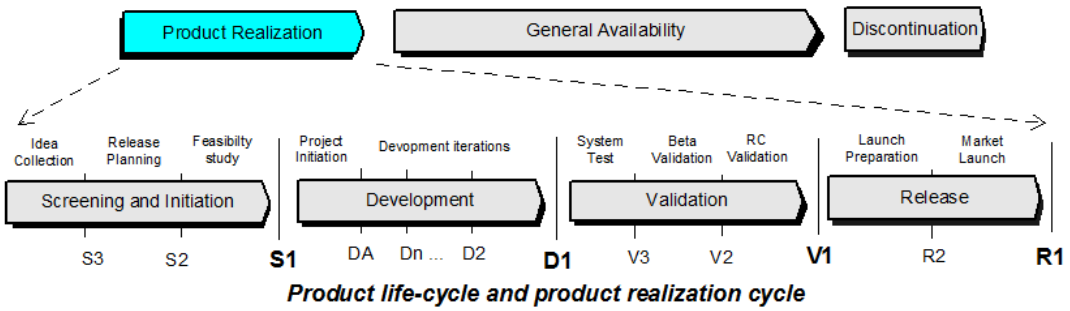


Figure 14: F-Secure FPRP from 1999[PP09]

In 2005, the RUP based FPRP 2.0 was commonly used within the company. FPRP 2.0 was an improved version of the original FPRP with more emphasis on the incremental development of software and services[PP09]. Before this, Agile methods were tried out in certain small research projects without major business importance. At that time, a project was launched to create a new kind of commercial product for small and medium sized businesses. The project team decided to start piloting agile methods, and this was the first time when these methods were used at F-Secure in a commercially-oriented development project.

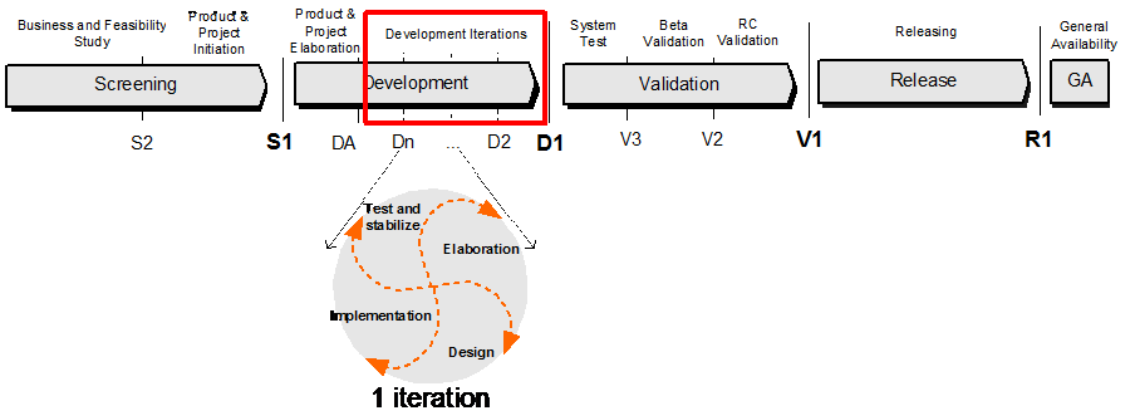


Figure 15: F-Secure FPRP 2.0[PP09]

Project Steering Group The Project Steering Group (PSG) was a forum for the project stakeholders to keep updated and have a say in the project decision making. Typically, the composition of the PSG could include various representatives from support, sales, training, documentation, marketing, R&D, and always the project manager, chief quality engineer and the product manager. The main responsibility for the PSG was to approve the project scorecard metrics, development and validation (beta, RC and RTM) milestones[PP09], and after the project give the project a score according to the pre-set scorecard.

Project Manager The Project Manager was a R&D position and a strong internal role that drove the project efforts towards the next release. Together with the Product Manager, Project Manager was the operational lead for the continuously ongoing R&D release projects.

Product Architect The Product Architect was senior level position responsible for a product at the module integration level and inter-team coordination of development efforts.

Chief Quality Engineer Chief Quality Engineer was a senior quality engineer position responsible for tracking and communicating the overall product quality level and coordinating quality and testing practice improvement initiatives among quality engineers.

Team Leader The Team Leader position was a manager position for a team representing line management to the team. Team Leader was also the team representative for the Project Manager in projects.

Component teams The teams were aligned with the architectural structure, one team focusing on the long-term development of one or several related software components. Typically, a component team composition included a software architect, some developers and several quality engineers.

5.2.4 Benefits and Drawbacks

The product council practice had several benefits. It was a clear decision making body for approving projects and conflict escalations. Having a lot of decision making power, it could flexibly balance investments between different business areas. The meetings covered the current status and roadmap reviews for the future projects, and thus one could easily get a good picture of the R&D activities around the company. The project approval practices required early commitment from R&D for the contents, so once a project was approved, all the stakeholders, such as sales and marketing, knew exactly what to expect as the outcome.

While the portfolio decision making had a high level of agility, it had a lot of challenges. Investment allocations were short and mid-term, basically per project basis. As inter-project dependencies were not identified on time, priority and investment conflicts emerged frequently which overloaded portfolio management level with escalations. Long-term strategic decision making at this level was unstructured and difficult.

The program level was fragmented, lacking both coordination and direction. While the product manager was the key role at this level, potential synergies from cooperation between the product managers were not fully leveraged.

At a project level, early commitment to a fixed scope frequently forced schedule extensions beyond what was originally planned. Projects were often late by several months, which caused problems for other projects.[PP09] Component teams were contributing to several projects at the same time, causing a lack of focus and significant task switching overhead. Piloting could be conducted only at the late stages of the project and quality feedback from customers and partners was only received late, when they had access to the software.[PP09]

5.3 2006 – Company-wide agile adaptation and creation of product management function

5.3.1 Portfolio level – No changes

The Product Council remained unchanged as the forum for portfolio level decision making.

5.3.2 Program level – Product management function

To support product managers' co-operation, all of them were grouped into a separate Product Management function, where the product managers had product or product line responsibilities, focusing on the "product" part of the four P's. The product managers owning a product line were acting as team leaders for the group of technical product managers, who were responsible for the application level, or the detailed feature composition and definition of the product functionality.

5.3.3 Project level – Scrum and XP adoption

A company-wide decision was made to move to Agile based software development processes. Teams related directly to the same product lines were collected under the same organizational units, and other development teams were grouped similarly to reflect the technological domain areas.

The Scrum based F-LEX process was created to replace the previous FPRP, and product backlog was introduced as the primary tool for the Product Manager. The development teams were still based on architectural components and started maintaining their own *Area backlogs*. These were owned by the team and maintained by the software architect. One month sprints were used as the R&D heartbeat. Each sprint preceded several heavy iteration planning meetings, and at the end of the iteration there was a massive demo session that covered the accomplishments of all the teams for the past iteration.

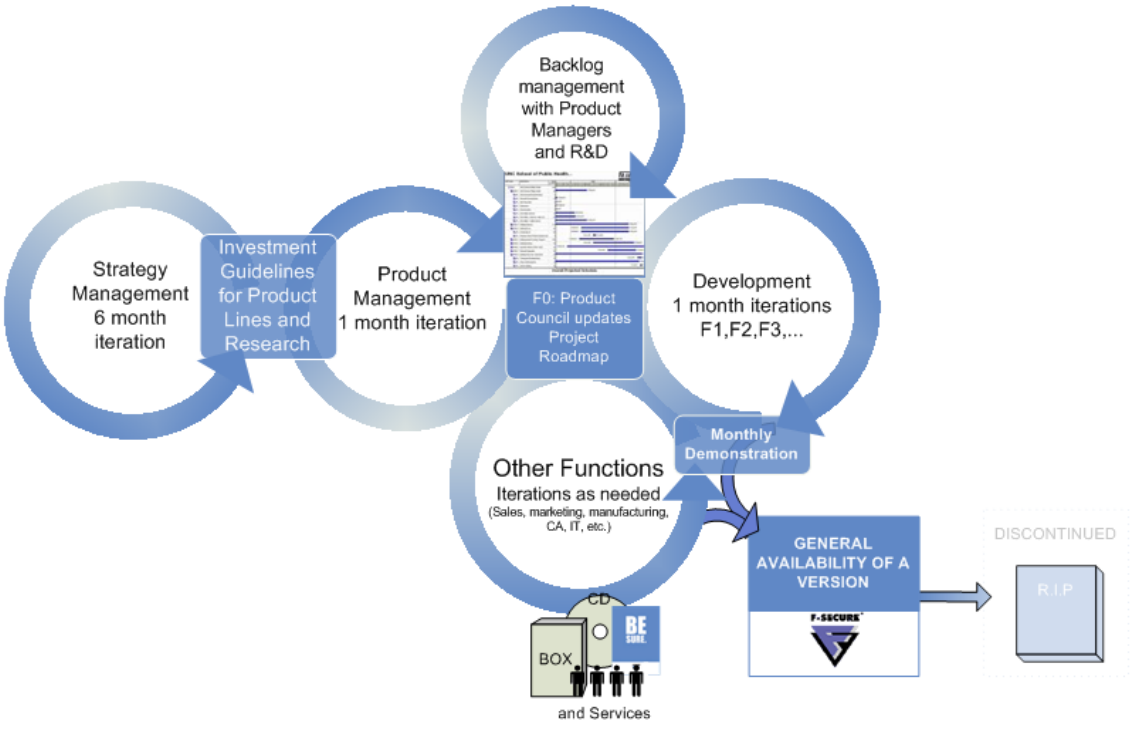


Figure 16: F-Secure F-LEX[PP09]

FPRP (and RUP to an extent)	F-LEX (and Scrum to an extent)
Role & document focused	Team and customer value focused, deliverable-oriented
Heavy emphasis on testing during validation	End-to-end system testing during every iteration
Committing to the scope early in the process	Committing initially to themes & minimal viable scope, final detailed scope through the iterations
Fixed resources and scope translating flexible schedule (=missed deadlines)	Fixed cadence & fixed resources; flexibility in scope (but also in time in rare cases, if deemed necessary)

Table 3: Biggest differences in FPRP and F-LEX[PP09]

5.3.4 Benefits and Drawbacks

The centralized product management function enabled better cooperation between product managers at the program level. However, product management was overloaded with day-to-day tactical work leaving insufficient time for long-term strategic thinking.[F-S07]

The use of agile methods on the project level had certain positive changes: the process framework was now truly iterative, and the project scope management was focused.[PP09] On the other hand the weekend overtime "sprints" had to be used as an extension to resourcing.[PP09] Still, projects were often late although not as badly as before, but continued to have a cascading effect on the other projects. Also, other departments working with R&D did not change their practices nor adapt to the agile way of working[PP09], causing friction and difficulties in cooperation.

5.4 2007 – Introduction of solution management

5.4.1 Portfolio level – No changes

The Product council remained unchanged as the forum for portfolio level decision making.

5.4.2 Program level – New solution management function

A solution concept was introduced to challenge the established narrow product definition. The term "product" was previously used within the company to mean software packages, documentation plus support and maintenance services. The term "solution" was introduced to cover one or several of these products on a *augmented* and *potential* product portfolio level, focusing on the customer's total *consumption system*[Lev80, KK06]. For this, a separate solution management function was formed, which had a team of *Solution Managers*, each owning the vision, the business

plan and long-term roadmap for several years ahead for the total offering portfolio for their selected customer segment. The customer segment was defined by essentially grouping existing related products together as different kinds of product portfolios. Solution management and product management functions were separated, even though solution and product manager peers were co-operating. Still, despite the conceptual separation the daily work of solution and product managers was very much related and overlapped with each other.

The operating domain of product management and the relationship with solution management, with guidelines or recommendations for balancing the focus of activities and stakeholders was clarified with better job description, which are summarized in the Figure 17, Figure 18 and Figure 19. Still, the day-to-day duties exceeded the desired share of working time.

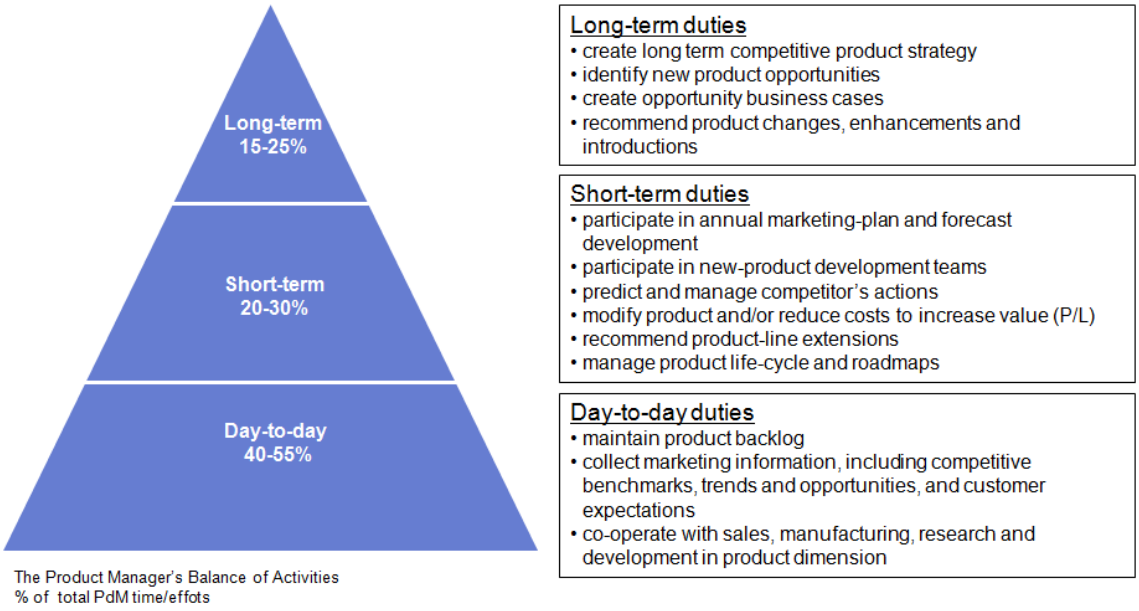


Figure 17: F-Secure product management balance of activities[F-S07], adapted from Gorchels[Gor00]

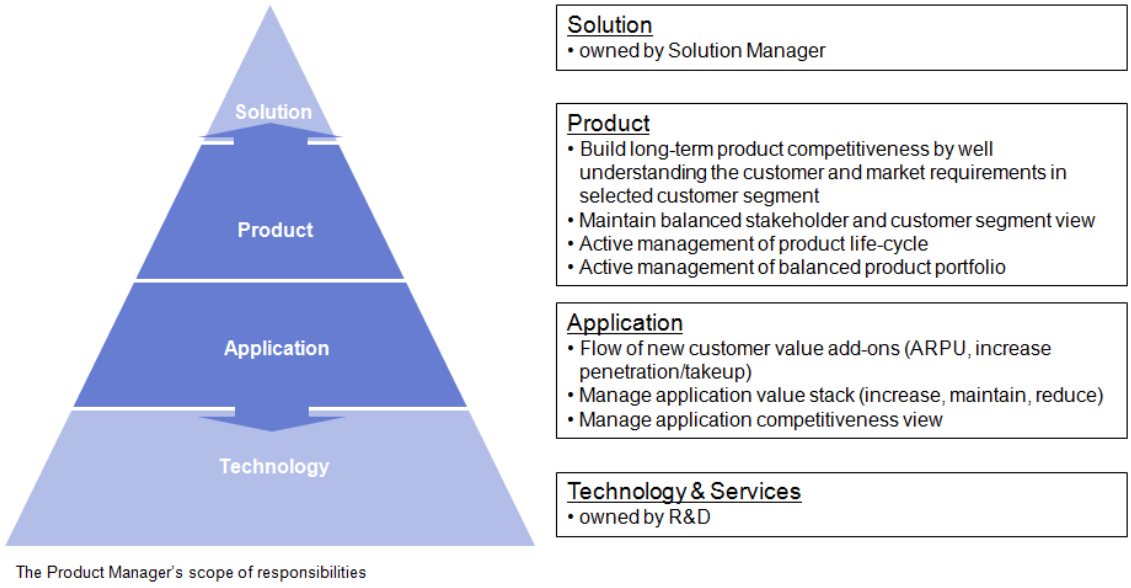


Figure 18: F-Secure product management domain[F-S07]

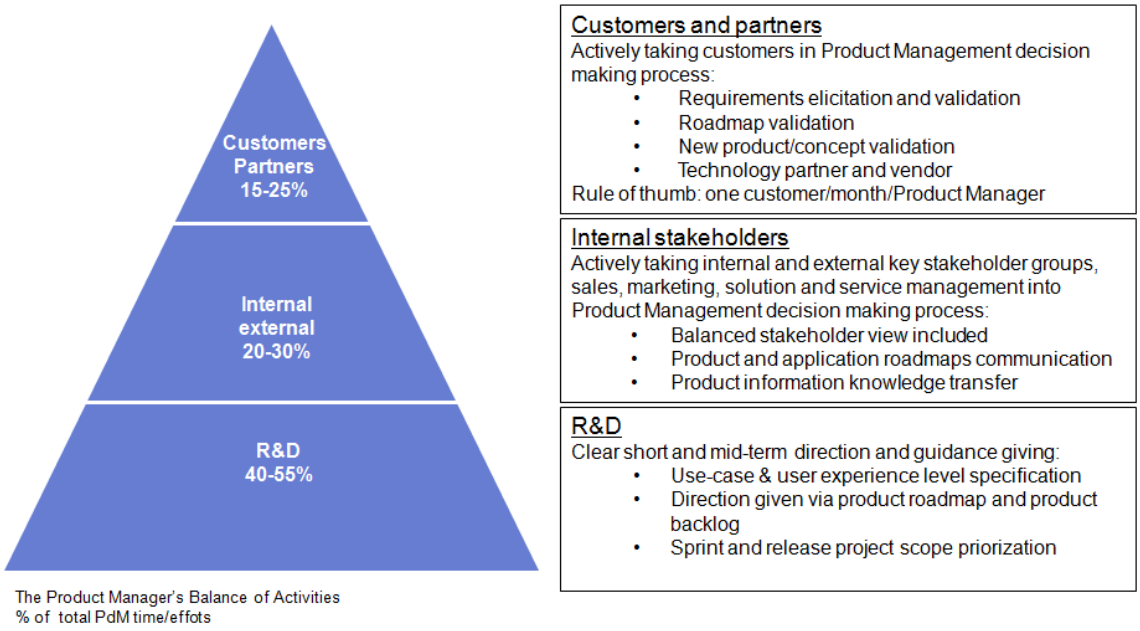


Figure 19: F-Secure product management stakeholder scope[F-S07]

5.4.3 Project level – Agile practices applied

The R&D continued to operate according to the Scrum based F-LEX process and used the Scrum tools and practices in an applied manner. The applied ways were following selected agile and Scrum practices, but not the principles. Due to communicational and visibility problems and component based team structure, completing

a feature from end-to-end required multiple costly handovers, and several problems arose from communicational misunderstandings.

5.4.4 Benefits and Drawbacks

The implementation of solution management provided more resources for product line planning at the program level. It started to focus on the long term planning that was previously getting too little attention from the product management function. Still, the product management function was overloaded as the responsibilities between the solution management and the product management were unclear, and the whole "solution" concept was undefined.

The project level agile adaptation was facing several challenges. Intimate understanding about project goals and objectives were not at a sufficient level from the beginning of a project, and thus it was difficult to get any commitments beyond the next iteration. Active scope management was simply a continuous reduction of the scope, which caused frustration from other departments as R&D would not "commit to anything". On the process side, the difficulties of scaling Scrum to efficiently synchronize and coordinate large multi-site projects were also discovered.[PP09]

The front end planning was seen bad by "religious agilists", that made creating large, completely new systems painful. Long-term architecture was not coordinated but it just "emerged", which caused growing architectural debt to start building up. The focus was too much in Scrum implementation and not enough attention was paid to the engineering disciplines, causing quality debt to start building up.[PP09]

5.5 2008 – Customer value stream and feature team transformation

5.5.1 Portfolio level – Renewed product council

The heavy product council practices and responsibilities were streamlined: the information sharing part of the council agenda was dropped, and the product council was explicitly defined to be the forum for approving the starting of projects only. The composition of the product council members was limited to a smaller number of people, all of whom were directly involved in project planning or in decision making. These changes successfully changed the previous heavy meeting days into a single day, and later half day efficient decision making meetings. The maintenance and approval of roadmaps were dropped. The roadmap work was previously duplicated in 1-year product roadmaps by product managers and long-term solution roadmaps by solution managers. Now there were only the solution roadmaps, which were reviewed in 6 month strategy cycles. The difference between the solution and product roadmaps was that solution roadmaps with release themes were confidential even inside the company, while product roadmaps focusing on a shorter timescale and feature sets were used internally for activity planning of different supporting

functions, and were typically published to external sources as well.

5.5.2 Program level – Product management function discontinued

At the beginning of 2008, the separate product management function was discontinued, and the product managers were moved into R&D; their positions were renamed as Product Owners, to support the original Scrum ideology. This transition was part of a bigger re-organization, that served the purpose of streamlining the feedback loop from customer-facing services to product development and back. Internally, the theme of the transition was named as "the line of sight", reflecting the customer focused value chain. In this value chain, the product owner and the solution manager were the catalysts in the middle, solution manager acting as the interface for the customer facing services, and product owner for R&D.

5.5.3 Project level – Feature teams

Soon after the product management function discontinuation, also the R&D internal structures were reorganized, from the specialized component and architecture based team structure to cross-functional "feature teams" with mixed skill sets. The vision for the feature teams was to cut the interdependencies between team silos and enable the independent end-to-end responsibility of value added development work within a team. Feature teams were working with one feature at a time, and once a feature was complete, the team could switch to any other team which was on top of the release backlog. Each feature team had a dedicated scrum master who was a team leader of the previous organization or a senior member of the previous development teams.

When the new feature teams started to figure out their role and new responsibilities, they started to require more guidance from product owners. While there were several teams working with a single product release at a time, there was only one product owner for the release, and few supporting product owners that owned their own related products in the product portfolio. These supporting PdO's were waiting for their turn to drive the development effort of multiple teams, and were willing to assist with another release in the mean time. This changed the previous R&D and product management collaboration model completely: PdO's were involved with multiple teams all the time with continuous planning, demo and reprioritization and day-to-day decision making tasks, which were previously mostly handled by the lead developers of the component teams. Even though supporting PdO's were there, the decision making was not delegated and was under the control of the single PdO in charge at that time. The teams' decision making escalation threshold was low and the result was overload on the current release's product owner. While the development teams got day-to-day guidance, the rest of the product owner's stakeholders got less bandwidth, and there was no time for strategic long term planning that was previously product manager's responsibility.

Eventually the day-to-day decision making scalability issues were resolved by implementing an *Area Product Owner* role or a *Product Owner Proxy* role for some teams working on some specific feature area. While the other Product Owners joined the demo sessions of these teams for providing feedback, the Area Product Owner was the dedicated specialist for the team, with the delegated responsibility for the team's backlog prioritization, and for day-to-day decision making and user story elaboration. The Area Product Owner role was not usually a full-time position. The investment for the specific feature area was defined in rough release planning by how much calendar time a dedicated team could use for the feature area or the release theme. Together with the investment guidance and a shared vision of the goals, the Area Product Owner could carry on with the team independently.

5.5.4 Benefits and Drawbacks

The renovation of the portfolio level decision making codified product council as the decision maker for official project approvals only. While this made the product council mission clear, other problems on this level were not addressed but moved outside the product council. The information sharing nature of the product council meeting was dropped, and the lack of roadmap reviews reduced internal cohesion. The portfolio level decision making was still only short and mid-term prioritization between projects, not between business areas. In general, communication between the different decision-making layers was not smooth, priorities and investment levels were not understood correctly, and agreed investment levels were not respected for lower priority projects.

The value chain mapping with the "line of sight", and better specification of the solution manager's role clarified the responsibility split between strategic product line focused solution management and product owners at a program level. The product owners were now closer to the teams and communication between the R&D teams and product owner(s) got better, which increased commitment and involved more people in innovation. This however made the project level scalability of the product owner role a problem, and the product owner for the release was constantly overloaded. The end-to-end responsibility of a feature team reduced handovers, task switching and communication overhead and increased the focus on the project level. Still, the links between decisions and items on backlogs and roadmaps on different levels were not always evident nor explicitly stated.

5.6 2009 – Agile F-Secure 2.0

The linkage between different layers of decision making – more precisely the connection from a team's backlog item to the solution roadmap/backlog was seen as an important communicational improvement area. This problem was addressed by a more explicit definition of these decision making levels, and how these are connected, as visualized in Figure 20. The model was adapted from Dean Leffingwell, but adjusted to comply with company internal roles, terminology and artefacts.

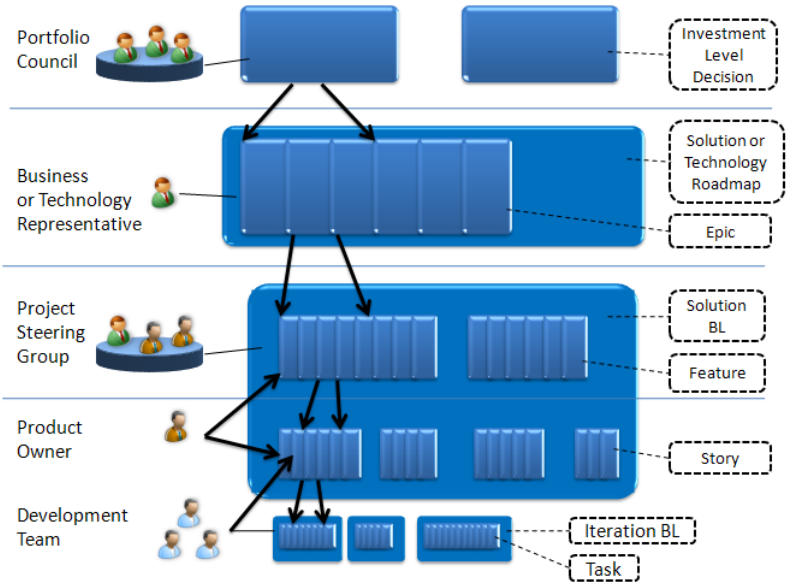


Figure 20: Different decission making levels at F-Secure[F-S09]. Adopted from Dean Leffingwell.

5.6.1 Portfolio level - Portfolio council

At a portfolio level, the portfolio council was formed from the responsible ET members and the VPs representing the different business areas, with the mission of solving problematic investment allocations between businesses. The work of this council was later assigned to be led by a dedicated portfolio manager.

As the new decision making unit made explicit investment decisions between different businesses and product lines, the importance of former product council diminished to be a formal bureaucratical step for official project approvals, which was quickly replaced by business unit internal decision making forums. These business unit steering groups took the responsibility of portfolio management and conflict resolution inside the dedicated investment allocation for the whole business unit.

5.6.2 Program level - Business units

In the next organizational iteration, the solution management function was discontinued, new business units were formed and the Product Owner position was again re-defined. The former Product Owner position was retitled as Technical Product Manager, while the Product Owner was defined as a role within a project, to reflect the dual roles in a matrix organization. The Technical Product Managers were scattered, either to report to R&D engineering, under the same function as the release management team and the development teams, or to the business unit together with the peer Solution Managers. This split affected the focus and orientation of the position to divert either more time towards the R&D or business, due to managerial influence and daily communication: the technical product managers' desks were ei-

ther located with the release management team or with the relevant business unit management team.

Product ownership in the F-Secure organization at the time of writing is a collaborative team effort. The team consist of experts with complementary skills and knowledge. The key characteristics and the current desired position mapping is shown in Figure 21. If we map them to the Pragmatic Marketing Framework^{TM}, we will see that the explicitly defined responsibilities of the roles cover certain areas of the framework, as shown in Figure 22.

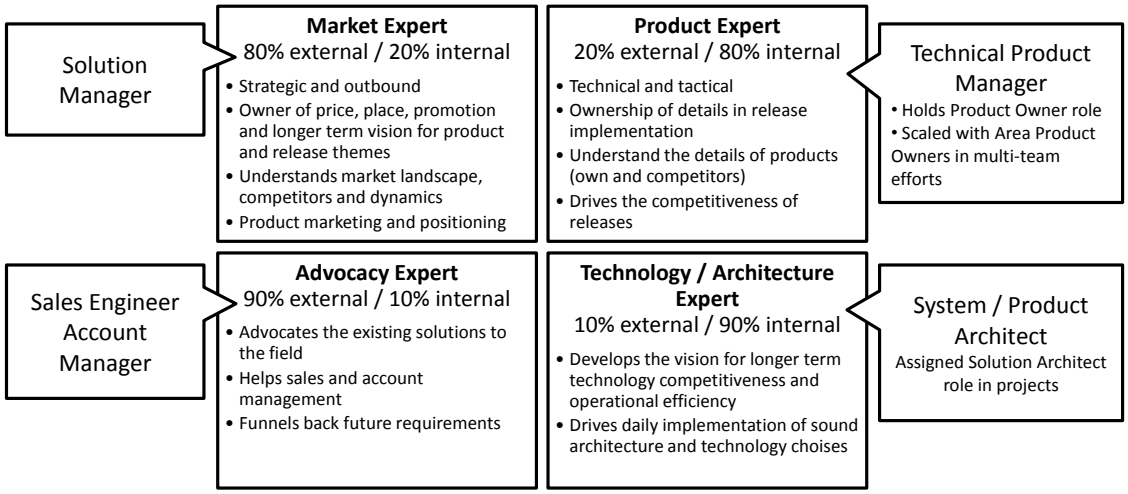


Figure 21: Requirements management roles at F-Secure Corporation[F-S09]

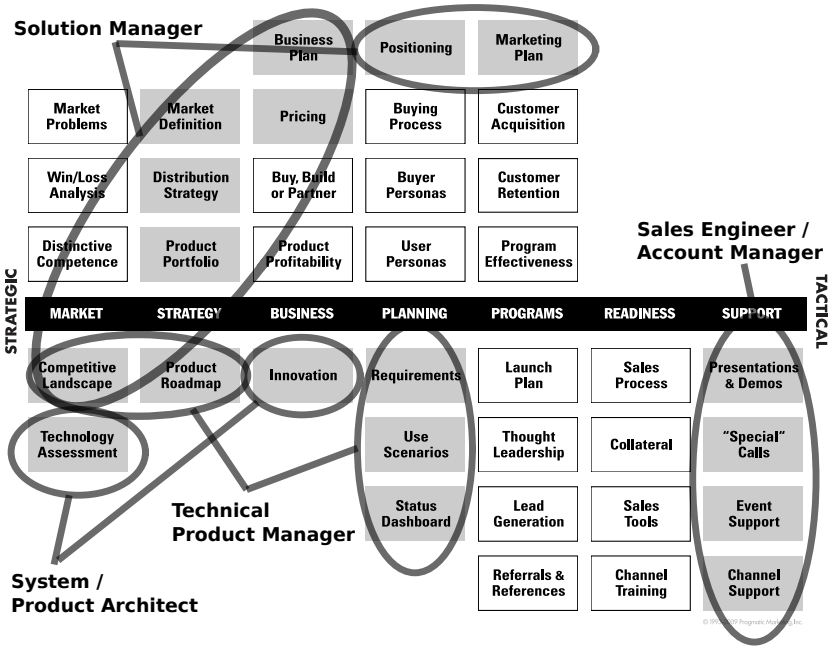


Figure 22: Requirements management roles at F-Secure Corporation, mapped to the Pragmatic Marketing Framework^{TM}

While many of the areas are not explicitly covered by the requirements management roles, many of these activities are still covered by some of the defined roles or some other role within the company. Two other roles that are not covered in the requirements management role quadrant of Figure 21 but are needed to fill most of the gaps in Figure 22, are the *Marketing Manager*, the *Channel Manager* and the *Technical Services Manager*, who represents the *Sales Engineers* and *Account Managers*.

Solution Manager The Solution Manager is a product management role that is considered the market expert of his/her domain, and they have the strategic business development responsibility over the solution product portfolio. The Solution Manager 'owns' the business over a solution that includes one or several products that form a product portfolio to a selected customer segment that is served through a selected sales channel. The Solution Manager is thus responsible for market definition, business plan, pricing, positioning, distribution strategy, etc. for several products and services, which together create the solution.[ET09]

Although the explicitly described responsibilities in the Figure 21 form a scattered map, it is evident that the activities in the business-strategic upper left corner of Figure 22, should implicitly be part of the Solution Manager's work. These include for example discovering market problems, win/loss analysis, leveraging the organization's distinctive competence, buy or build or partner decisions and product profitability.

Competitive landscape analysis is shared with the Technical Product Manager, where the Solution Manager focuses on competitors at a portfolio and positioning level. While the Solution Manager is a marketing oriented role, the marketing plan related work is actually led by the dedicated Marketing Manager.

Solution manager has also a role within the project:

> *"The Solution Manager keeps the PSG informed of market conditions. The Solution Manager also acts as a primary driver and coordinator in solution-related issues. This includes understanding the market, forming the solution vision, creating business and launch plans and coordinating between different functions. He chairs the PSG and assigns business values to Solution BackLog items. He is an important driver of Solution BackLog prioritization work."[F-S09]*

Technical Product Manager Technical Product Manager is the technical product, customer and area expert that 'owns' the technical implementation of the product and its component features. A Technical Product Manager serves as the reference source for describing product related functionality both in non-technical as well in technical terms for supporting internal and external stakeholders. The Technical Product Manager is the central person bridging business and technology stakeholders and translating and communicating objectives and limitations.

Technical Product Managers work together with the Solution Manager to define the solution roadmap, influencing the long term direction of the solution and defining the product vision. The Technical Product Manager owns individual products and releases (software, services and processes) as part of a solution and is responsible for benchmarking the technical competitiveness of the product against competitors.

In development projects, the Technical Product Manager takes the role of the Product Owner, owning the product backlog and acting as the customer proxy for the development teams.

Technical Services Manager Technical Services Manager is responsible for the technical competence of the technical sales engineers globally. This role provides training and assistance for product knowledge, and gathers and communicates aggregated feedback to the solution manager and technical product manager from product sales. The Technical Services Manager is very outbound oriented, assists sales engineers & account managers in special cases, and is responsible for technical sales tools, presentations and event support. A Technical Services Manager does not operate with specific products, but with the whole corresponding offering for the served customer segment. When positioning this role in the Pragmatic Marketing Framework™, it is clearly in the technical-tactical corner of the map.

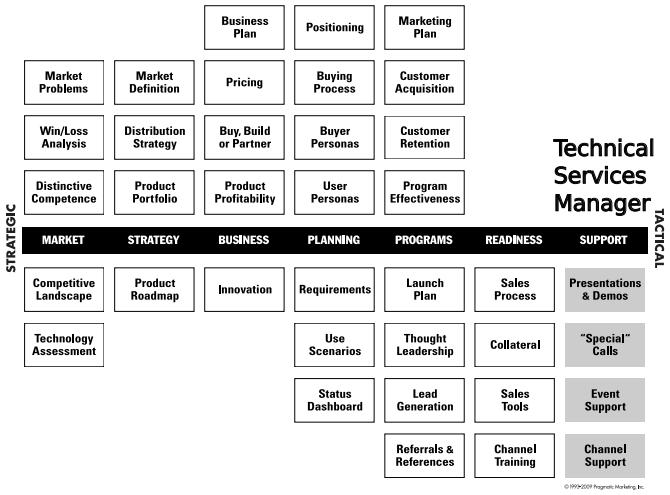


Figure 23: Technical Services Manager responsibilities mapped into Pragmatic Marketing Framework™

Marketing Manager Marketing manager is responsible for forming the marketing messaging and communications, creating marketing and launch plans and driving product launch activities. This role also makes sure that collateral is created and marketing communication happens and is aligned correctly. The Marketing manager is mostly involved in the post-project activities when the product release was prepared for the general availability, and not really in the R&D product development project.

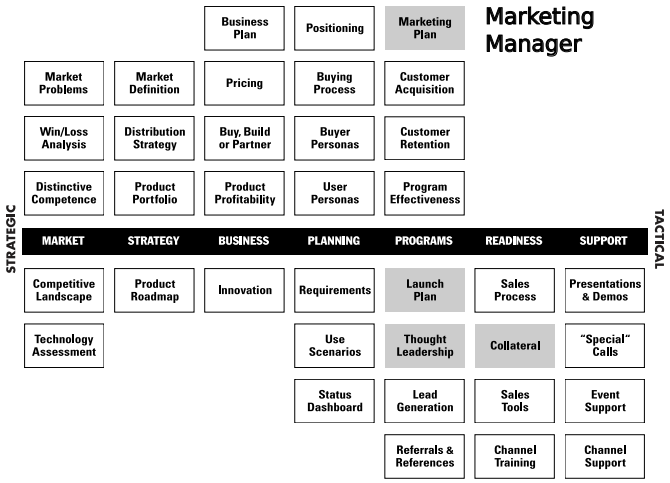


Figure 24: Marketing Manager responsibilities mapped into Pragmatic Marketing Framework™

Channel Manager The Channel Manager was responsible for the business of a specific sales channel, such as the sales that were made by the reseller partners. The Channel Manager is the person in charge of the sales channel effectiveness and sales process, and creates and enforces training, channel support and tools, customer and partner acquisition/retention programs to improve and accelerate the sales performance.

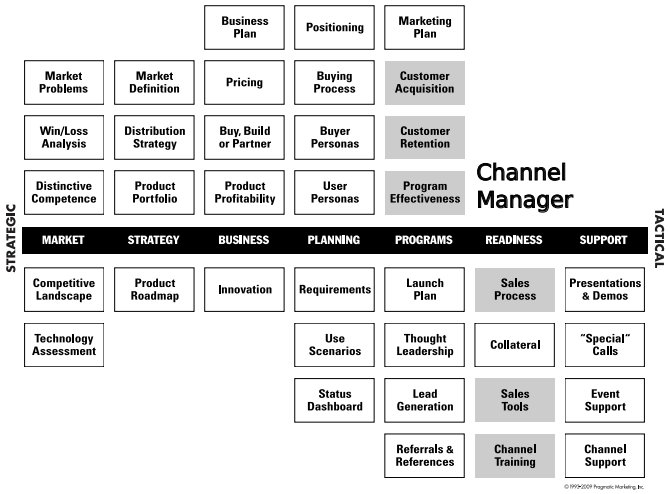


Figure 25: Channel Manager responsibilities mapped into Pragmatic Marketing Framework™

5.6.3 Project level - Agile 2.0

As the original F-LEX process framework description was outdated and no more reflected the way product development projects operated, a new F-LEX 2.0 process framework definition project was started with the goal of describing the re-usable and working practices that had evolved and proven useful since the first F-LEX, with certain generalized high-level models based on theoretical research for areas which were seen unclear or dysfunctional due to being without a well working internal implementation. As a step forward, F-LEX 2.0 stated explicitly the F-Secure software development principles which guide the selection of working practices. In the context of a company's iterative operation, the focus of F-LEX was limited to cover the repeated cycles within a release project as shown in Figure 26. The new F-LEX 2.0 is intended to be continuously updated, to add, modify and remove parts whenever there is a need. This new internal process framework at the time of creation was based on Dean Leffingwell's view of a scalable Scrum process flavoured with XP and some internally-developed best practices and lean & agile values adjusted for the business and company's cultural and decision making context. While F-LEX 2.0 was deliberately limited to cover the R&D side of the project and process work, it also included some high-level theoretical models covering the product ownership structure of the company. The profile of the project steering group was raised as the top level decision making unit for the project, as shown in the Figure 27.

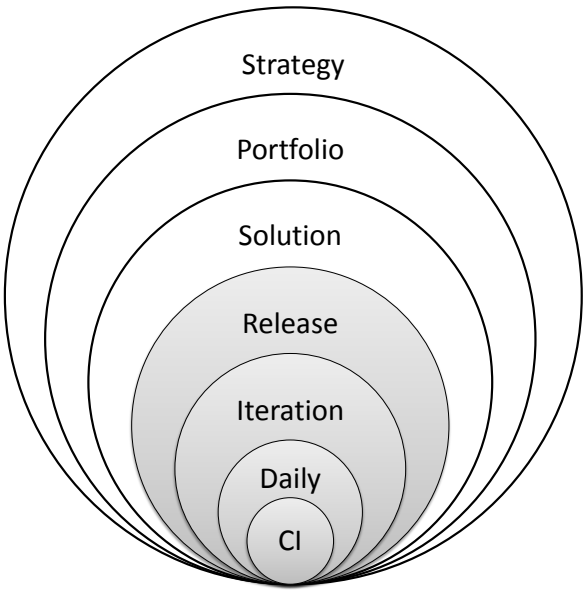


Figure 26: Iteration levels at F-Secure Corporation. F-LEX 2 covers the grayed levels from continuous integration to the release-level.[F-S09]

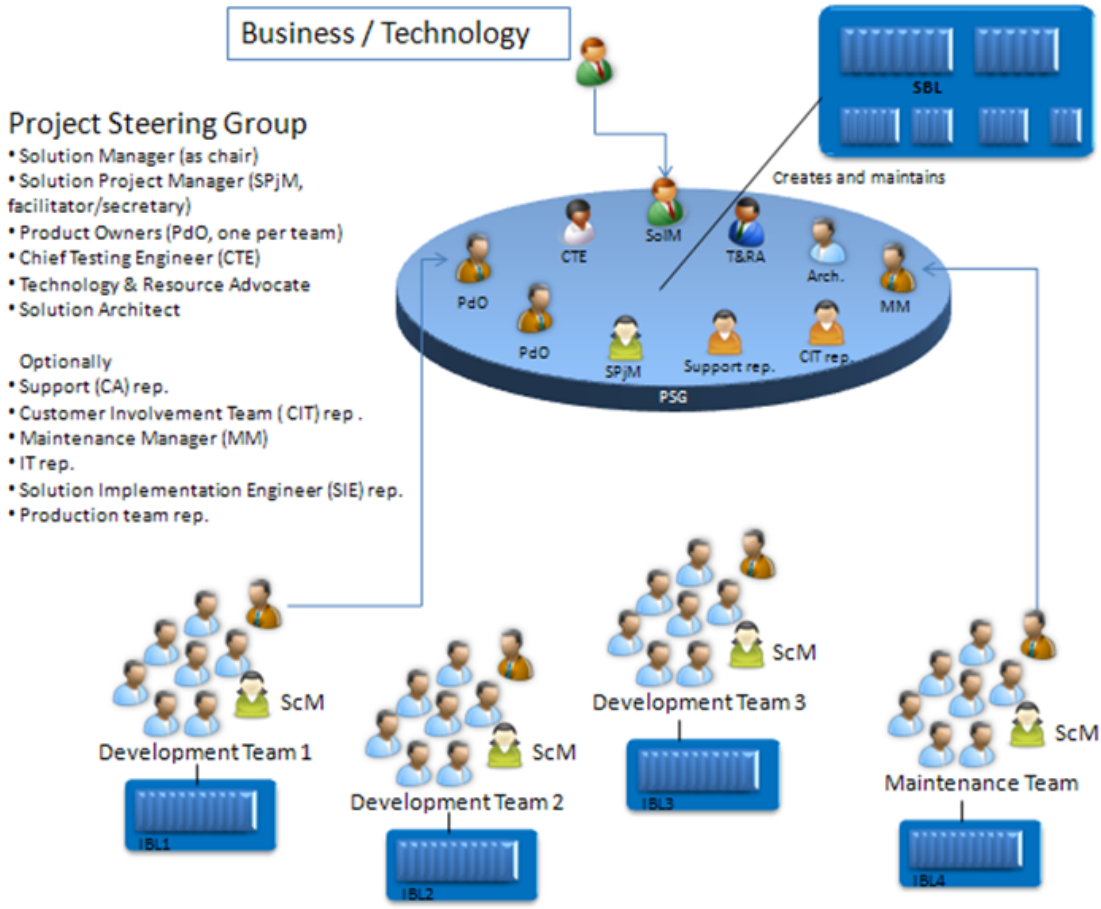


Figure 27: F-LEX 2.0 project organization in multi-team projects[F-S09]

The main roles in this multi-team project organization are described below.

Project Steering Group The Project Steering group's purpose was expanded from only approving milestones and scores into being an active participating unit for final priority calls and providing guidance and decisions for major changes throughout the project:

> *"Project Steering Group is the primary working decision-making body in the project. The mission of the PSG is to run a project that produces maximum value to the company in long-term, given the restrictions of epics, time, and resources. The Project Steering Group defines and monitors the project's measurable objectives and success criteria. It also monitors and reacts to changes in the costs, schedule and scope of the project and approves Release Candidates (RC) and Public Betas."[F-S09]*

In addition, PSG has the responsibility over the solution backlog:

- *PSG is responsible for maintaining the correct priorities in the Solution Backlog*

  - *– Priorities need to be revised at least before iteration planning activities*
- *PSG is responsible for including all necessary work in the Solution Backlog in the form of features and stories*
- *PSG is responsible for staying informed of project progress*
- *PSG may organize its' activities in the way that it finds most suitable*

[F-S09]

Product Owner The product owner is not a permanent position but a role in a development project. The product owner can be one person or a shared responsibility between multiple area product owners. Area product owners focus on a specific product feature or requirement area, in case there are multiple development teams working simultaneously on different development areas or features. However, there is only one dedicated product owner for a team at a time, even though a product owner can serve multiple teams at the same time. In other words, the relationship between a product owner and a team is one to many, not many to one.

> *"The Product Owner (with PSG) maintains a prioritized list of Features and Stories with the help of stakeholders. The Product Owner owns and prioritizes the Stories in the PSG. The Product Owner owns the results of the work done in the iteration, including quality."[F-S09]*

The main purpose of the product owner is to act as the interface between the development team(s), solution management and other stakeholders, and to prioritize the stories in the team's iteration backlog. The product owner owns the product or area backlog the team is currently working on. He or she selects which user stories are added to the backlog and sets their relative priority order based on business value versus estimated effort ratio analysis, deciding which of them the team starts working on next. The product owner is responsible for maximizing the delivered value, and ensuring that the functional and non-functional requirements are met. The product owner also clarifies the backlog items to the teams, and re-scopes, splits and re-defines backlog items to be clear and small enough so that the team can properly estimate the required effort and start working on them throughout the project. More precise product owner responsibilities according to F-LEX 2.0 – with a note that there may be multiple product owners for a solution – are listed below.

- *Acts as an interface between development team(s) and solution management, and other stakeholders.*
- *Is responsible for the Solution Backlog and maintains the full backlog (at least for main releases) in priority order (at least at the highest priority level). This provides visibility. In multi-team projects, Product Owners need to work together to form the Solution Backlog, coordinated by the Solution Project Manager.*

- *Knows and actively communicates market opportunities and the competitive situation, works closely with the solution management.*
- *Is an active member of the Project Steering Group (PSG), works closely with the development team during iterations.*
- *Consults and negotiates with the development team, PSG, PdC, and stakeholders. Resolves conflicting Solution Backlog item priorities.*
- *Is responsible for the results of the iterations, including quality. Must approve any compromises on quality. Estimates with the team if it makes sense to sacrifice quality for features and communicates this to everybody related to the project.*
- *Makes proposals for release contents to the Solution Manager. Informs about scope changes or calls for release (whenever valuable additions are in the product).*
- *PdO clarifies the Solution Backlog items to the development team.*

[F-S09]

Solution Project Manager Solution project manager is the person who follows the overall progress of the project and co-ordinates communication between the different teams and stakeholders, driving collaboration beyond organizational boundaries. A Solution project manager works together with the scrum masters solving problems related to the processes, practices and removing impediments. As the organizer and facilitator, the solution project manager makes sure activities such as the solution backlog prioritization work, solution demos, milestone planning happen on time. They also monitor the project status, schedule, scope and resourcing and keeps the project steering group informed.[F-S09]

Solution Architect Solution architect was a similar role to the previous product architect, facilitating and coordinating inter-team technology and architectural issues, but with increased responsibility over the input of architecture development stories to the backlog.[F-S09]

Chief Test Engineer Chief test engineer as the solution architect also had new responsibility for providing input for the backlog, and acting as the product owner for *system team*, for managing the infrastructure development. Chief test engineer's main focus is to measure the quality of the overall solution and ensure the project steering group and the project group awareness of its status.[F-S09]

Technology & Resource Advocate This role is similar to the program manager's position as the *resource owner*, but with a possible input to the technological vision for the content. The resource advocate is a person who is able to make resourcing decisions for the project and possibly can have responsibility over the technical competitiveness.[F-S09]

Scrum Master Scrum master is a team based coaching role for the practices and removing impediments, in an orthodox Scrum manner.[F-S09]

5.6.4 Benefits and Drawbacks

With business unit separation on both the program and project level, portfolio level investment levels were changed from vague guidance to explicit definition that enabled and empowered the business units to have proper authority, capability and the resource pool to build uninterrupted long term business and product development plans without resource competition between business areas. This change supported the removal of obstacles from long and medium term planning, as now the authority over priorities of investment was at a program level. Inter-project dependencies and conflicts could now be solved more efficiently and without heavy escalations.

The recognition of a product management team working structure with four roles was a step towards a generic definition of a specialist team that covers the essential areas of the product management domain at a program level. Whilst the direction of this move is good, the model that was used is still immature and it does not include some important areas and roles of the product management domain. As the ownership of some strategic product management areas is only implicitly defined, there is a risk that these areas are overlooked or poorly prioritized.

The connection between program level and project level ownership is not yet properly defined in F-LEX. The roles that operate on both program and project levels are the solution manager, the technical product manager, and the solution architect. The project level ownership is shared by the members of the project steering group, in which all these roles are represented in addition to the other internal project stakeholders. As the solution manager is the chair of the project steering group, it would imply that this role has a bigger responsibility with regards to the project success. The technical product manager also has the product owner role serving one or several teams, and he or she works with other product owners when there is a large number of teams involved. The project level virtual team, like the project steering group may be sufficient for project execution but the inconsistent organizational placement of these roles either to report to R&D or to the program level business unit management, especially the technical product manager role, may not support long term product management development and team work. Despite these challenges, F-LEX 2 is a significant improvement over past project models with explicitly stated context-sensitive high level software engineering principles and frameworks, even though it is still incomplete and ambiguous on these areas.

6 Conclusions

In this section, we conclude the study by seeking answers to the original research questions presented in the subsection 1.1 Problem Statement. The focus of the answers is in the large scale product development in an enterprise organization context. The questions are listed below, with the answers found during this research.

What is the role of the product manager?

Product manager is a commonly known role used within organizations with commercial products. Although there is no clear definition of a product manager, this role can be defined by the area of responsibilities that are typical for this kind of profession. A product manager is a person who is working within the product management domain, that can be visualized, for example, with the Pragmatic Marketing Framework^{TM}, that is shown in the Figure 4 in subsubsection 3.1.2 Product Management Domain on page 11. For a selected product or product line, there can be one person or several people with specialized skills operating within this domain. The pragmatic marketing framework^{TM}is not the only one which tries to describe the characteristics of different kinds of product manager roles, but does a good job by covering the multi-dimensional aspects and the dynamic team work model of the program management layer of a modern product development company. When using Leffingwell's 3 layer portfolio-program-project decision making levels that are used in the theoretical synthesis, product manager(s) operate on the *program decision making level*.

What is the role of the product owner?

The product owner is a role defined in the Scrum framework to embody the content decision-making authority and to abstract out related context sensitive work. In many environments, this covers the job of several individuals, possibly even departments. As Scrum was created from organizing the work of a single team, and scaling that one up, the product owner is the role serving the needs of a development team. The explicitly defined product owner responsibilities in Scrum list the key needs of R&D to organize, plan and operate rapid value delivery. The product owner is the operational decision-making role with leadership characteristics defined at the *project level*, that elaborates requirements, educates the team, and formulates and then communicates goals. In addition, the product owner is the role that links the program and project decision making levels, and maps the collaboration of R&D to the product management domain.

Is the product owner the same as product manager?

As pointed out in previous sections, the product manager and product owner roles exist and operate on different levels and focus on a different timeframe, thus *the role*

definitions are not the same. However, the product owner for the project should be part of both program and project levels, in order to leverage synergies and ensure communication between these two. The product manager, or a specialized product manager when product management domain responsibilities are distributed between several people, *should be* working as the product owner to ensure proper communication and collaboration:

> *"For commercial development, the Product Owner may be the product manager. For in-house development efforts, the Product Owner could be the manager of the business function that is being automated."[Sch09]*

In the suggested product management domain splits using the Pragmatic Marketing Framework™, these responsibilities fall into the area that is typically covered by a technical product manager. When the product owner role is scaled with supporting product owners, the team's product owner *does not have to be* product manager to fullfil his or her responsibilities for the team:

> *"The Product Owner can be a Team member, also doing development work. This additional responsibility may cut into the Product Owner's ability to work with stakeholders."[Sch09]*

How does the Scrum product owner scale up, and how the responsibilities can be shared between several persons?

The key problem in scaling product ownership appears to be about defining the decision making structure, which is left undefined in Scrum. The research of this study shows that there is no single best way to organize the responsibilities of product management and ownership. There are several active writers in the agile community with their own views of best practices ranging from ideologic to pragmatic, industry-driven alternatives.

In larger organizations focusing on commercial product development with product portfolios consisting of multiple products or product lines, Leffingwell suggests three decision making levels:

- *Portfolio decision making level* for defining long term investment levels between products and product lines
- *Program decision making level* for mid and long term strategic product planning over several release cycles
- *Project decision making level* for planning and tactical decision making happening within a project

Based on this study, we can conclude that such a structure can be extended by adding more levels above the portfolio decision making level, which define investment levels between – for example – business units, or also more abstract investment areas in larger or more complex organizations. The scope of the study limits our focus to the program and project levels. Inside these two levels there are different ways to scale and share the work.

The program level maps to the traditional product management domain. The wide spectrum of activities on this level requires a combined set of different domain specialities for full coverage. These activities are well described and visualized in the Pragmatic Marketing Framework^{TM}, which maps them to a strategic-tactic and business-technology oriented 2-dimensional axis. Hohmann et al. suggest scaling operations within this domain by splitting the work among specialized product managers, each focusing on activities that are closely connected, i.e. form a connected area in the framework. Hohmann et al. give several examples to share the product management domain differently between one to four roles. We conclude that the domain can be split with the same logic among a larger number of more specialized roles, as long as the roles can co-operate as a team.

Project level is where the traditional project management and Scrum fits, and where the development teams operate through project planning and execution. Thus, this is also the level where the Scrum product owner role is defined. When only a few teams are involved, a single product owner is enough and scaling is not needed. When the number of teams exceeds the capacity of a single product owner, there is a need to find solutions to share the work. Larman and Vodde suggest first pushing more responsibilities to the teams before looking for other alternatives. It is important that the authentic information source is as close to the development teams as possible, to avoid misunderstandings and information degeneration as it passes through a chain of multiple people. When this approach is not enough, more product owners are needed to properly serve the needs of the teams. In the section 4 Current Discussion we found support for product owner team structure, with the product owner and supporting product owners. There are currently no guidelines how to share the work within the product owner team, but when following lean thinking the supporting product owners should possess fully-delegated authority over the area they are working on, to keep the decision making as close to the teams as possible.

However, the culture and the size of the organization affect what works and how – the same solutions do not similarly apply to a project which has 5 teams or 50 teams. In different contexts and organizations with different maturity levels, different areas are more important or problematic. Context awareness was also highlighted by Mary Poppendieck and Ola Ellnestam in their presentations at ScanAgile 2009 conference[Pop09, Ell09]. One should find no reason to be dogmatic, but try to learn to see what principles are the most relevant and valuable for the context. Based on this analysis one should then choose to use the documented *practices*, or create your own, to *support the principles*.

How a product owner and/or product manager role is implemented in a large scale agile product development?

In this study, the product ownership structure of F-Secure Corporation's windows clients program was observed over the period of several years during which agile methods were taken into use. The organization, the maturity of the used processes and role responsibilities have evolved in an iterative manner by solving the biggest problems one by one, and discovering and adapting to new challenges. The case study shows empirical support for Dean Leffingwell's 3 decision making layers during the observed time period – product council, portfolio council and business unit steering groups operating on the portfolio level, product management and solution management functions, and later selected roles from the business units operating on the program level, and R&D project organization operating on the project level. When we compare the latest situation with the product ownership synthesis presented in subsection 4.6 Product Ownership Synthesis, we see some level of correlation. The most differences occur at the program level, where product management responsibilities are split between specialities, but not as granularly as suggested by the selected theoretical model in the synthesis, leaving some responsibilities only implicitly defined. Thus it is possible that these responsibilities do not receive the desired attention. Re-defining the product management domain split may provide some areas for future improvements. Also, the definition of the connection between program level and project level ownership is left quite vague – by the F-Secure F-LEX 2.0 this connection is made with the Project Steering Group which has the Solution Manager as the chairman. Project level ownership is scaled with supporting product owners as suggested by the synthesis. These roles have similarities to the area product owner role suggested by Larman and Vodde, while the clear project lead is a shared responsibility. Overall, as a real-world implementation example it seems to be a context-dependant mix with slight deviation from the theoretical synthesis. This may provide, as we anticipate, improvement possibilities in the case study organization.

7 Discussion

The proposed strengths and weaknesses of this study are gathered below, with the suggestions for further studies for the continuation of the research.

7.1 Weaknesses of the study

Overall, the scope set is challenging – focusing on the scope for theory or for a case study would have allowed deeper analysis, but would be challenging for a master's thesis as sources are currently very limited. A wider scope would have ensured good source availability, but would limit the analysis to a very general level and would not have provided sufficient challenge. The selection of sources for the different viewpoints in the section 4 Current Discussion can be argued to be incomplete, as more discussion on these topics is happening during and after the writing of this analysis. It is possible that there are contradictory findings that challenge the validity of the results that are based on the represented and thus analysed viewpoints.

The case study provides only a single example of the real world implementation of product ownership and analysis of other cases would give better confidence for the generalization of problems and found solutions within the product ownership domain. In addition, the dual role of the writer as the subjective actor within the case and objective researcher documenting and analyzing the case raises questions about subjective bias in the analysis and conclusions, as a lot of the case study is written based on personal experience.

7.2 Strengths of the study

As for the merits of this study, we see that the research questions were covered successfully. Despite the limitations of material availability, the theoretical part of the study provides a good general coverage of the software product manager and product owner roles. The study gives several examples and principles for splitting the responsibilities on different levels and forms a synthesis of several product management and ownership views and terminology.

The case study is spread to cover a history of evolution over several years. This study provides value to the researched organization in several ways. Documentation of the evolutionary history of product ownership and analysis of the benefits and drawbacks of the changes provide opportunities and support for further organizational learning. Also, the analysis of the current responsibilities gives guidance on the open issues within the product management and product ownership domain that need more focus, thus pointing out next possible improvement areas.

7.3 Further research

Based on this study, the areas that could be topics for the further research are listed below.

- Are there dependencies to the portfolio management level that affect how the work should be organized?
- How do requirements management tools and practices scale between different decision making levels?
- Analysis of other case studies with different context, size of the organization and maturity level, to verify the validity of the theorethical analysis and synthesis presented in subsection 4.5 Product Ownership Analysis and subsection 4.6 Product Ownership Synthesis. How these different aspects affect the results, i.e. are the same problems and solutions found?
- Can the scrum model be scaled to cover the whole enterprise context?

References

[AWSR03] Pekka Abrahamsson, Juhani Warsta, Mikko T. Siponen, and Jussi Ronkainen. New directions on agile methods: a comparative analysis. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 244–254, Washington, DC, USA, 2003. IEEE Computer Society.

[BA04] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2 edition, 11 2004.

[BBvB+01] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development. http://agilemanifesto.org/, 2001. Cited: 18.7.2009.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, USA, 1999.

[Bec99] K. Beck. Embracing change with extreme programming. *IEEE computer*, 32(10):70–77, 1999.

[BM01] Eric Bonabeau and Christopher Meyer. Swarm intelligence: A whole new way to think about business. *Harvard Business Review*, 79(5):106–115, 2001.

[CLC04] David Cohen, Mikael Lindvall, and Patricia Costa. An introduction to agile methods. *Advances in Computers, Advances in Software Engineering*, 62(66):2–67, 2004.

[Cop09] James O. Coplien. Why lean and agile are in conflict, and how to resolve the conflicts. October 2009. Presentation at ScanAgile 2009, 15.10.2009.

[Dru74] Peter Ferdinand Drucker. *Management: tasks, responsibilities, practices*. Harper & Row, 1st edition, 1974.

[DS90] Peter DeGrace and Leslie Hulet Stahl. *Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms*. Prentice Hall, 1990.

[Ell09] Ola Ellnestam. Context over dogma. 2009. Presentation at ScanAgile 2009, 15.10.2009.

[ET09] F-Secure Solution Manager Esa Tornikoski. Interview, May 22nd 2009.

[F-S07] F-Secure Corporation. Product management job description, 2007.

[F-S09] F-Secure Corporation. F-lex 2 – common policy and guideline for f-secure r&d projects and software engineering. Version 2.1.5, October 2009.

[FH01] M. Fowler and J. Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

[FHGM09] Peter Firstbrook, Arabella Hallawell, John Girard, and Neil MacDonald. Gartner magic quadrant for endoint protection platforms, May 2009.

[Gal09] Robert Galen. *SCRUM Product Ownership – Balancing Value From the Inside Out*. RGCG, LLC, 4 2009.

[Gor00] Linda. Gorchels. *The Product Manager's Handbook: The Complete Product Management Resource*. NTC Business Books, Lincolnwood (IL), 2000.

[Gra95] Ian Graham. *Migrating to object technology*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1995.

[Hig01] Jim Highsmith. History: The agile manifesto. http://agilemanifesto.org/history.html, 2001. Cited: 19.7.2009.

[HJM08] Luke Hohmann, Steve Johnson, and Rich Mironov. Living in an agile world: the strategic role of product management when development goes agile. *The Pragmatic Marketer*, 6(5):4–10, 2008.

[Int09] International Rugby Board. Laws of the game - rugby union 2009. http://www.irb.com/mm/Document/LawsRegs/0/IRBLaws2009ENlores_7685.pdf, 2009. Cited: 1.8.2009.

[Joh08a] Steve Johnson. The strategic role of product management. http://www.pragmaticmarketing.com/strategic-role-of-product-management/Strategic_Role_Product_Management.pdf, 2008.

[Joh08b] Steve Johnson. The strategic role of product management webinar. http://www.pragmaticmarketing.com/strategic-role-of-product-management/, 2008. Cited: 19.7.2009.

[Kha08] Saeed Khan. Agile/scrum software development and product management. http://onproductmanagement.net/2008/04/29/agiledev_and_pm/, April 2008. Cited: 29.12.2008.

[KK06] Philip Kotler and Kevin Lane Keller. *Marketing Management (12th Edition)*. Prentice Hall, 2006.

[Kot99] John P. Kotter. *What Leaders Really Do*. Harvard Business Press, 1 edition, 3 1999.

[Lar03] Craig Larman. *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley, 2003.

[LE07] Mike Lowery and Marcus Evans. Scaling product ownership. *AGILE Conference*, 0:328–333, 2007.

[Lef07] Dean Leffingwell. *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley Professional, 1 edition, 3 2007.

[Lef09a] Dean Leffingwell. Scaling software agility – the blog: Best practices for large enterprises. http://scalingsoftwareagility.wordpress.com/, 2007 – 2009.

[Lef09b] Dean Leffingwell. Agile requirements – role of the product owner. http://scalingsoftwareagility.files.wordpress.com/2009/11/ch-11-role-of-the-product-owner-rev-11.pdf, November 2009. Cited: 16.11.2009.

[Lef09c] Dean Leffingwell. The big picture of enterprise agility (rev. 2). http://scalingsoftwareagility.files.wordpress.com/2009/08/the-agile-enterprise-big-picture-2.pdf, August 2009. Cited: 19.8.2009.

[Lef09d] Dean Leffingwell. The big picture of enterprise agility, updated graphic. http://scalingsoftwareagility.files.wordpress.com/2009/01/picture-1.png, May 2009. Cited: 22.08.2009.

[Lef09e] Dean Leffingwell. Biography of dean leffingwell. http://www.leffingwell.org/bio.html, 2009. Cited: 17.9.2009.

[Lev80] T. Levitt. Marketing success through differentiation – of anything. *Harvard Business Review*, 58(1):83–91, 1980.

[Lik03] Jeffrey Liker. *The Toyota Way*. McGraw-Hill, 1 edition, 12 2003.

[LV08] Craig Larman and Bas Vodde. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley Professional, 1st edition, 12 2008.

[LV09] Craig Larman and Bas Vodde. Lean primer. http://www.leanprimer.com/downloads/lean_primer.pdf, 2009. Cited: 12.9.2009.

[LV10] Craig Larman and Bas Vodde. *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Addison-Wesley Professional, 1st edition, 2 2010.

[Mar09] Ryan Martens. Rally software agile blog: Agile and lean software development – an oxymoron? http://www.rallydev.com/agileblog/2009/06/agile-and-lean-software-development-an-oxymoron/, June 2009. Cited: 4.10.2009.

[Mir08] Rich Mironov. Revenue products need product managers, not product owners. http://www.enthiosys.com/insights-tools/pm-prod-owner/, 2008. Cited: 12.9.2009.

[Pit93] Matthew Pittman. Lessons learned in managing object-oriented development. *IEEE Software*, 10(1):43–53, 1993.

[Pop09] Mary Poppendieck. Leadership and self-organizing teams – incompatible or synergistic? 2009. Keynote presentation at ScanAgile 2009, 15.10.2009.

[PP03] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.

[PP06] Mary Poppendieck and Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, 2006.

[PP09] F-Secure CTO Pirkka Palomäki. Agile transformation at F-Secure. Presentation at OO Days at Tampere University of Technology, 9.12.2009, December 2009.

[Pra09] Pragmatic Marketing. Pragmatic marketing framework™. http://www.pragmaticmarketing.com/pragmatic-marketing-framework, 2009. Cited: 24.08.2009.

[PU09] F-Secure Director of Product Management Pekka Usva. Interview, November 11th 2009.

[Rat03] Rational Software. Rational unified process – best practices for software development teams. http://www.ibm.com/developerworks/rational/library/content/03july/1000/1251/1251_bestpractices_TP026B.pdf, July 2003. Cited: 20.1.2010.

[SB02] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*. Prentice-Hall, Upper Saddle River (NJ), 2002.

[Sch95] Ken Schwaber. Scrum development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 117–134, 1995.

[Sch07] Ken Schwaber. *The enterprise and scrum*. Microsoft Press, Redmond, WA, USA, 2007.

[Sch09] Ken Schwaber. Scrum guide. http://www.scrumalliance.org/resource_download/598, May 2009. Cited: 19.7.2009.

[Scr09a] Scrum Alliance. Scrum alliance coaches. http://www.scrumalliance.org/community/coaches, 2009. Cited: 01.08.2009.

[Scr09b] Scrum Alliance. Scrum alliance members. http://www.scrumalliance.org/community/members, 2009. Cited: 01.08.2009.

[Scr09c] Scrum Alliance. Scrum alliance trainers. http://www.scrumalliance.org/courses/trainers, 2009. Cited: 01.08.2009.

[Tik08] Ari Tikka. Viewpoints to agile product management. Agile Finland Seminar, Helsinki, March 4th 2008.

[TN86] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard Business Review*, 64(1):137–146, 1986.

[Yin94] Robert K. Yin. *Case study research: Design and Methods*. SAGE Publications, Thousand Oaks (CA), 1994.

Appendix A: Agile Principles

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity – the art of maximizing the amount of work not done – is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[BBvB+01]