HAAGA-HELIA
University of Applied Sciences

Master's Thesis

Improving requirements management practices in agile software development environment

Antti Kokko

Master's Thesis
Degree Program in Information Systems Management
November 2013

HAAGA-HELIA
University of Applied Sciences

Abstract

15.11.2013

Information Systems Management

Author or authors Antti Kokko	**Group or year of entry** 2012
Title of report Improving requirements management practices in agile software development environment	**Number of report pages and attachment pages** 64 + 9
Teacher(s) or supervisor(s) Jouni Soitinaho, HAAGA-HELIA	

Requirements management is an essential part of software development. This work emphasizes product management's role in a software development environment. Product management represents customer to product development. By turning development projects towards an agile requirements management through communication and training, the project was a success, delivered on time but with challenges, namely resistance to change.

In this case study, by utilizing the action research strategy, qualitative data was gathered using a comprehensive questionnaire. Altogether 36 professionals took part in interviews, results were analyzed and an internal development project roadmap was developed. One of the target projects played an important pilot role during the internal development project. In this particular pilot project, changes were implemented into practice with a help of a project manager, leading architect and with a product manager who was also the mentor of this thesis work.

Continuous development of practices improves quality and at the same time operates as a reminder for maintaining existing good practices. The study indicated that better process quality can be received with relatively small changes in practices. By utilizing and implementing requirements hierarchy and a structured way of handling even large amount of requirements has been shown to improve management work. This also helps other stakeholders get a better understanding of products content. Consistent and harmonious stack of requirements are easier to manage. Collaboration and continuous communication reduces ambiguity.

A key finding of this study is that requirement's continuous, systematic and collaborative management improves requirements quality. Better requirements quality with improved project practices are key factors in successful software development projects.

Keywords
Requirements management, Product management, Requirements hierarchy, User story, Agile software development

Table of contents

Abbreviations

Alusta	Platform powering Basware solutions
ASD	Adaptive software development
BET	Basware executive team
IA	Invoice automation
ID project	Internal development project
ICT	Information and communication technology
ISO	International organisation for standardisation
NG	Next generation
P2P	Purchase to pay
P-gate	Product development milestone
PLC	Product life cycle
PMT	Project management team
R&D	Research and development
RUP	Rational unified process
SA	System analyst
UA	User assistance
UI	User interface
UX	User experience
UML	Unified modeling language
WCAG	Web content accessibility guidelines

1 Introduction

Requirements management is a combination of understanding specifications, teamwork, and management. It is co-operation with customers, customer representatives and other relevant stakeholders. Software research and development is involved in this process and benefits from a structured, well-organized and clear requirements. Broadly speaking, requirements management can be seen as one of the core areas in an organization's development of software applications.

As processes, practices, and methods inevitably develop and change, requirements management in absolute terms is less effective - continuous development is needed. Identified problems and challenges should be addressed so that development projects can be carried out in a stable environment and to maximum effectiveness. In addition positioning an organization to be able to react to changes it is also important to maintain good working habits and methods. Software development should always aim for the best possible result by creating outcome, products or services that create real value for customers. The best requirements management practices should be utilized in software development projects and in larger programs. Identification, analysis and implementation of requirements management best practices help projects and programs successfully release products that are relevant for today's end user.

It is crucial for a successful software R&D organization to be able react in a timely manner when changes are identified and subsequently determined to be required. Organization must also aim to be flexible enough to ramp down or improve working practices which are evidenced to be ineffective or practices that prevent its key assets, its employees, from achieving their potential. After all, efficient and enjoyable work combined with achievement and the right incentives can be a very strong motivator.

In order to implement a successful development project, improved requirements management practices together with appropriate rationale that explains the cause of the changes should be communicated to relevant stakeholders with managed effort from the dedicated change agents within the organization. Finally, any new system should

consider how the new process might disrupt existing operations and provide solutions to mitigate resistance to change to the greatest extent possible.

1.1 Background

Basware's R&D unit identified that it had rather good but fragmented requirements management practices. Different methods were used in many various ways that disunited working practices across several projects and often led to inconsistency. Indeed, the movement of individuals between projects often led to ineffectiveness and dynamic development resources suffered from the present state the most and it was challenging for software developers and testers to maintain top performance levels.

In response, requirements management internal development project, later called ID project, was established in August 2012 by the management team of the Basware R&D unit. ID project team organized and started its work officially in October 2012. the project team's main goal was to discover and to determine but also to ascertain and establish improved requirements management practices and working methods into agile software development environment. To be able to achieve this, one of the objectives was to find out a more efficient way of handling requirements in broad product portfolio because Basware was in the middle of moving from developing individual modules towards designing and producing a broad and comprehensive solution suite.

The ID project focused on requirements management process and it was executed in two subsequent parts at Basware R&D unit starting on 22 October 2012. The length of the whole project was estimated to be approximately one and a half years. The original roadmap theme was intended to deliver world class and benchmarked common practices to P2P projects. The objective was to create and support capable best practices for system analysts across the organization together with unified P2P requirement management practices guided by high quality P2P vision and definition. The second high level target was to further enhance existing best practices.

1.2 Exposition of the current state

Requirements management is a sub process in Basware's PLC model and it recognizes areas such as requirements gathering, documenting, analyzing, prioritizing, road mapping, and tracing. Requirements management practices are documented and available in the organization's intranet. A requirements management "cook book" exists and it was last time updated in the beginning of 2011. Requirements management process in practice is considered, if not equal, but close to requirements specification process.

Process documentation contains high level instructions for feedback and idea collection through different systems. Creation of high-level specifications as well as detailed specifications into the system in a consistent and structured way is clearly instructed. Acceptance and review practices for requirements are recommended and included as part of the documentation. Project level guidelines such as requirements implementation order, postpone and rejection activities are also included in the instructions. Dependency follow-up, requirements tracking, and requirements change control methods are also identified and explained. Product and release specific requirement baseline setting activities completes the documentation.

The purpose of requirements management at Basware is based on the following principles:

- Ensuring that Basware products meet external stakeholder (customer) needs
- Ensuring that Basware products meet internal stakeholder needs (sales, marketing, product marketing, consultants, support and product development)
- Documenting relevant external and internal stakeholder needs
- Documenting Basware solution (functionality, architecture, and non-functional aspects of products)
- Ensuring that needs and internal decisions can be traced up until released product versions
- Enabling efficient customer communication
- Enabling design, implementation and testing

The role of a customer is important, identified and included in the process documentation. Requirement and opinion collection, approved release content communication, and possible schedule change notification practices are good examples.

Requirements management process diagram and role specific activities are based on a traditional waterfall method. As seen in the Figure 1, some of the requirements management activities end when the release development starts at P2-gate. The complete diagram is available in Attachment 7.

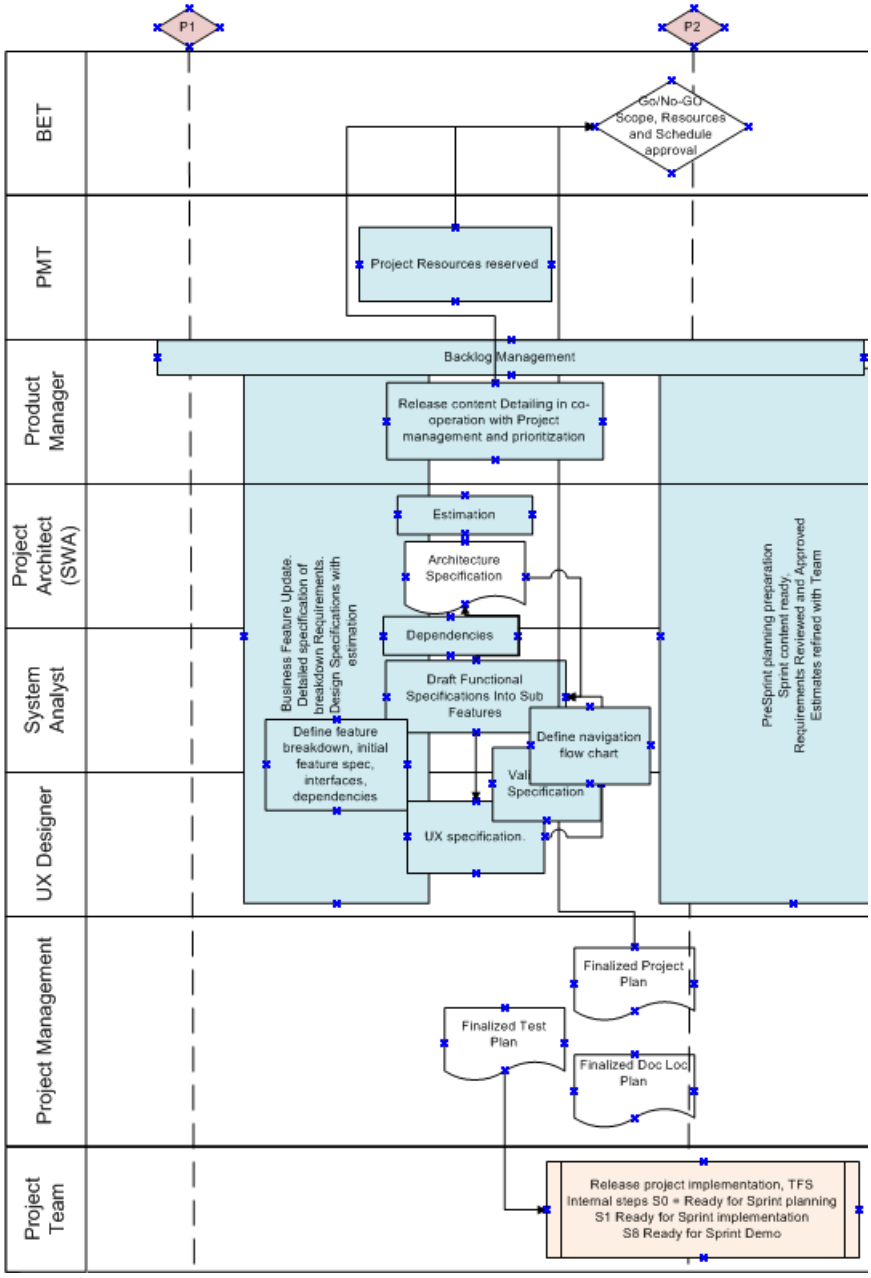


Figure 1. Snapshot of requirements management workflow diagram at Basware

1.2.1 General challenges

Sprint level practices are not similar between projects and a systematic way of working between projects does not exist at this time. Different requirement types and relations between them are not used similarly over projects. It is unclear how, for example, business features, use cases, features and other types or requirements are created, utilized, and managed. Requirements hierarchy is missing and even similar requirement types in Accept 360 requirements management application are not used in similar ways between projects. The chain from non-functional requirements to implementation is broken and it is challenging to transfer these requirements to everyday development and testing activities both on system and feature level.

Without exception, overall visibility to the current state of the development is missing. Product managers are too often bottlenecks for decision making and system analysts are not familiar enough with customers' initial needs and domain understanding is not in the best possible track. Responsibilities between product managers and system analysts are unclear. There is no process to determine which product requirements should be addressed as common requirements and how to do this. There is also a need for a channel to communicate and discuss these common or P2P level issues. Management and relevant stakeholders don't know how to turn business requirements into features. Product vision is not taken into account very well. Practices in sketching, innovating, planning and confirming plans together with other stakeholders varies significantly and responsibilities for decision making do not exist. The overall responsibilities on requirements management process are not well defined.

1.2.2 Unclear requirements management process in practice

Requirements management process is suffering from an "intra-release" PLC sub-process reputation, whereas it should be a continuous process guiding and resulting input to product development on a daily basis. Clear definitions for the whole process, terminology and requirement items are needed.

1.3 Research question and research scope

The purpose of this work was to explore requirements management practices in an agile software development environment. The research was conducted by utilizing the Action research strategy and was limited to Alusta program which consisted of two main product lines, Purchase and IA (Invoice automation). Program altogether employed 110 persons in 16 separate teams. The research project ended at the end of H1 2013.

The analysis of questionnaires resulted in a set of consolidated assumptions about problems that requirements management process has been dealing with. The more specific content of the research emerged during the second analysis round and finally formulated the research question:

How to transform requirements management process to better enable agile software development?

1.4 Motivation

Agile software development expects speed and accuracy from requirements management. Product management's role in requirements management process is essential to be able to create valuable products for customers at high quality. Basware's R&D organization set focus to the field of requirements management by establishing the ID project. The need of rapid improvements were identified.

Requirements management had also been a personal interest for this report's author several years. Once the author was selected as a member to the internal development project team, it became evident that thesis work would be part of the project. Motivated colleagues and a professional R&D community was in need of positive changes and the organization was ready to improve existing working practices.

1.5 Objectives

The overall goal of the study was to launch a series of improved requirements management practices across development teams working for the Basware Alusta program.

The first part of the ID development project focused on short term improvements which were subsequently implemented by the end of year 2012. The second part of the project focused on further improving existing practices but at the same time also identifying new efficient agile requirements management working methods which could be utilized and taken into use.

The first improvement areas were determined through the analyzing phase of the questionnaire results. The ID project team had originally planned to execute second questionnaire after the implementation based on the first questionnaire. Second round interviews were intended to cover larger group of different stakeholders from different units. Instead, the ID project team decided to concentrate once again on the results of the original questionnaire. The long term objective was to establish improved practices and methods for the Alusta program and its projects, teams and their members but also for other stakeholders working closely with development teams and product management. Selected approach helped the ID project team to better control the progress.

1.6 Internal development project team

The ID project team originally consisted of 5 persons. Project was overseen by the vice president for Basware's R&D procurement unit. Other team members were a system analyst representing the Invoice Automation unit, the lead system analyst representing common and non-functional requirements, and a senior process engineer representing tools unit. The author of this report, senior system analyst represented the Procurement unit, brought in the theoretical background for the project and was in a leading role during the implementation phase.

System analyst at Basware participates in product roadmap planning, product maintenance and product execution tasks. SA mainly creates and prioritizes functional requirements for product but also supports test case creation project planning. SA ensures that delivery of component, module or solution is in line to agreed standards, quality and design. SA analyses and identifies process defects and designs but also implements improvements.

ID project team defined and accepted best practices together and delivered these to R&D teams along the project. Project team size increased from 5 to 8 in the middle of the project. Two new members strengthened the team by bringing in project management viewpoints and new system analyst brought in cross-functional know-how which the project team was missing previously.

ID project team aimed to identify bottlenecks in processes and areas that management believed needed fixing. Main goals for the ID project team was to identify bottlenecks in processes, identify and correct ineffective practices, and finally improve organization's requirements management practices.

2 Research methodology

The strategy utilized in this study was Action research. The strategy in the requirements management ID project covered in the practice questionnaire, data analysis, conclusion and recommendation – this holistic approach involved -gathering, -implementation, and measurement executed by the ID project team. Action research was the most sensible and relevant selection for the project and when it was presented to the ID project team as a course of action, it was immediately endorsed by the team.

2.1 Action research

Argyris, Putman and Smith defines that action research may be appropriate when experiments are on real problems within an organization and are designed in a way that the outcome assists in a providing a solution. This involves an iterative process of problem identification, planning, action, and evaluation. If done correctly, action search could lead to re-education, changing patterns of thinking, and ultimately action. This is highly dependent on the participation of the focus group both in providing honest responses to questions and eventually in findings solutions that change corporate behaviour. It is intended to contribute both to academic theory and practical action." (Bryman & Bell 2011, 413.)

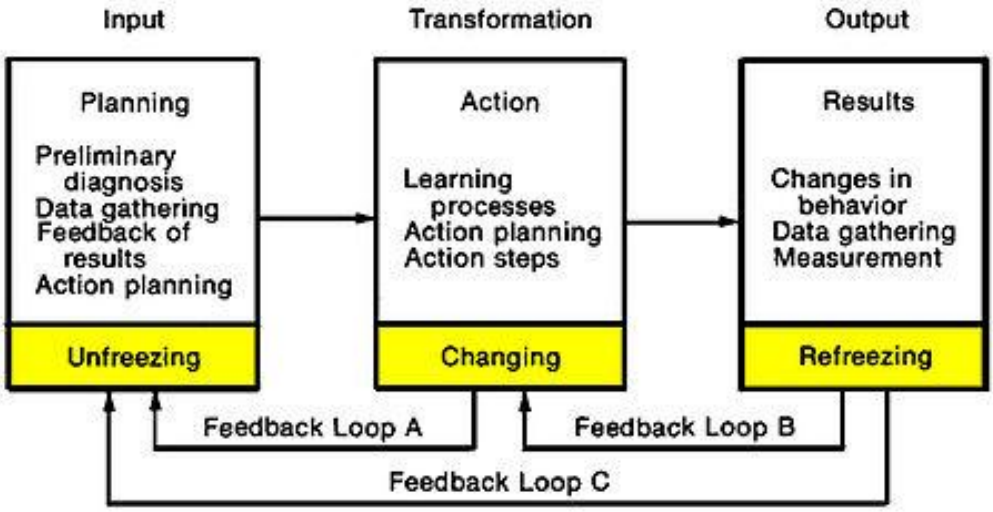


Figure 2. Systems model of action research process (Wikipedia 2006)

Figure 2 shows that the Action research cycle begins with a series of planning actions. Elements of this stage include a preliminary diagnosis, data gathering, feedback of results, and joint action planning. The second stage of action research is the action phase. This stage includes actions relating to learning processes and to planning and executing behavioural changes. Feedback at this stage would move via feedback loop A and would have the effect of altering previous planning to bring the learning activities into better alignment with change objectives. The third stage of action research is the output or results phase. This stage includes actual changes in behaviour (if any) resulting from corrective action steps taken following the second stage. Data are again gathered so that progress can be determined and adjustments in learning activities can be made. Minor adjustments of this nature can be made in learning activities via feedback loop B. (Wikipedia 2006).

Action research aims to make changes and improvements to a situation through a cycle or set of cycles of investigation, action and reflection, while at the same time reporting it in a way that is useful both to the project in hand and potentially to outsiders. Stemming from the work of Kurt Lewin in the 1940's, action research was originally a means of assisting people to move forward through enquiry into issues in their own lives. It has become widely used as a methodology for practitioner and collaborative research as it provides a straightforward way of taking a researching approach to practice or change. (Costley & Elliot & Gibbs 2010, 85).

Some forms of action research follow a cycle of four stages; planning, acting, or creating change; observing and data gathering; reflecting; and decision making. The 'plan-do-study-act' cycle used for business process improvement is a form of action research and similar principles are used in soft systems methodology. Ethical issues need careful consideration in action research due to the potential effect of changes on participants. (Costley & Elliot & Gibbs 2010, 85).

2.2 Agile utilization of action research

Method itself created a framework for the ID project and helped the team to develop in a structured way. By following the action research method model, ID project's work was divided into different stages. Planning phase of the project contained data gathering which was carried out in a form of several interviews. The questionnaire used in interviews is visible in Attachment 1. During the first planning phase, the ID project team created a short term plan and set its own targets for the reflection phase. Improvement ideas were recorded and results were measured subsequently.

Movement from the decision making phase back to reflection phase happened in the beginning of year 2013 when the second part of the project commenced. The ID project team once again redefined its objectives and progressed with an implementation plan. The analysis and discussion of the questionnaire results continued and helped the ID project team to focus on relevant areas. Results are gathered into Attachment 3.

Action research steps embedded to ID project's road map are displayed in the Figure 3. A selection of best practices and delivery to all P2P projects was the 2012 H2 theme, and further enhancement to existing best practices was the theme for H1 2013.

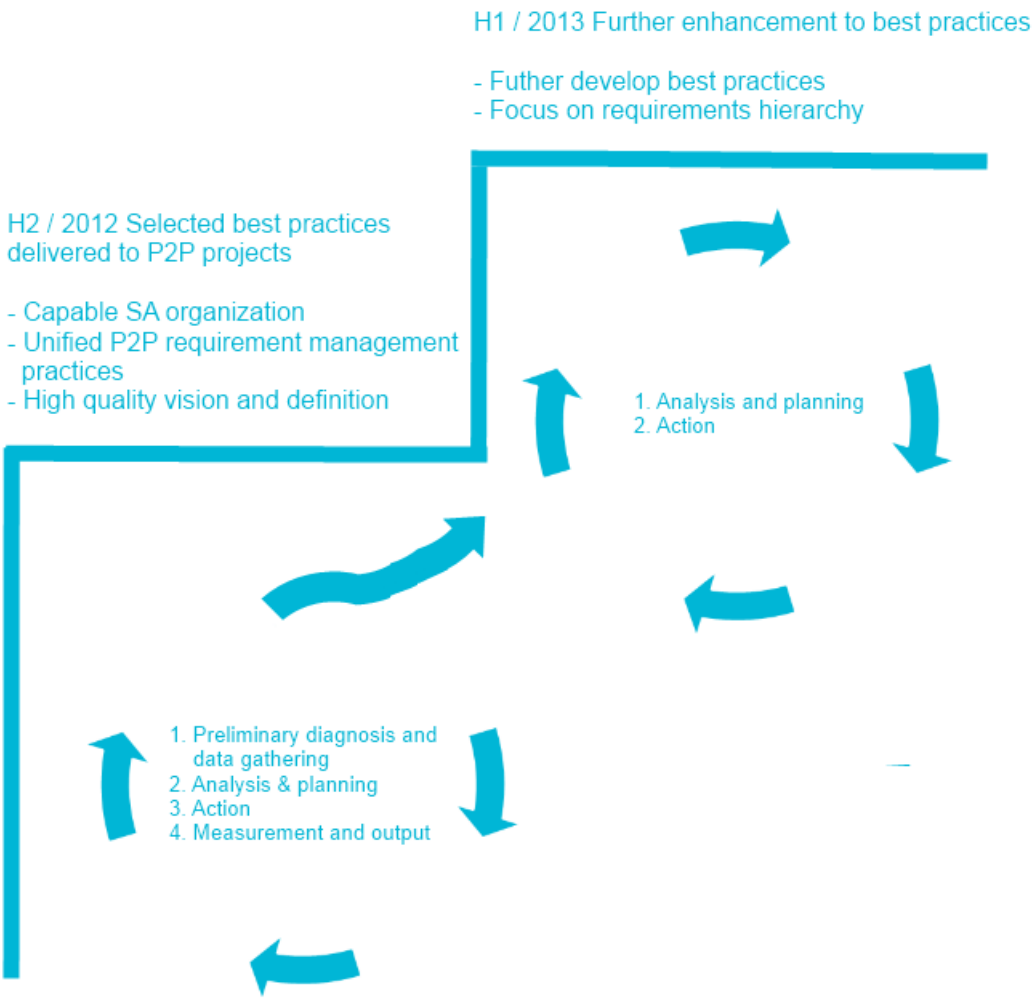


Figure 3. ID project roadmap and embedded action research steps

Basware's R&D product management team handed over the high level roadmap but ID project was given permission to later modify the roadmap during the project. The intention was to be able to reach more pragmatic and agile approach of the entire assignment. Initial roadmap guided the project team but it was discussed and changed few times as the project matured.

During H2 2013, ID project team created a questionnaire and selected a target group. The final amount of interviewees was 36. The roles of the interviewees are presented in Attachment 2. After a number of iteration rounds, the closed questionnaire received its final form and statements were categorized as appropriate. The senior process engineer on the team conducted the actual interviews on a one on one basis with the respondents. These in-person sessions took approximately 30 minutes each and once inter-

views were concluded and results collected, the ID project team analysed the data and prepared to take actions.

During the first planning phase, the ID project team also created a short term roadmap and set its own targets for the transformation phase, which was to come at a later time. Based on the new findings, the project scope was updated with more realistic objectives and action steps were agreed. Requirement management improvements related the work of system analysts in the first phase and a system analyst meeting was arranged where analysts were introduced to new best practices and trained by using material created by the ID project team. Measurement was done after a two month period through a series of follow-up meetings.

Movement from output phase back to transformation phase took place in the beginning of year 2013 when the second part of the project started. Due to general business disruption with the Alusta development program, the ID project team found it difficult to find time for common sessions and several meetings were either cancelled or held without a complete team in place. The ID project team struggled with different topics and future plans started to appear difficult to implement based on existing experience. During that time, one of the target projects presented an idea of requirements hierarchy and gave few initial reasons of how hierarchical requirement levels would help projects to manage their requirements more efficiently. Soon after this, ID project team decided to put all the effort to requirements hierarchy and started to plan and develop the concept further. The ID project team defined this to be its next objective.

Implementation of the requirements hierarchy was set to be delivered to all Alusta projects by the end of H1. The initial plan was to introduce and train the new requirements hierarchy to several groups including architects, product managers, UX and UA professionals, as well as system analysts. The purchase project team was the driving force during the research project and played an important role as a forerunner as new method and changes in practices were about to be taken into everyday work.

2.3 Theoretical framework and story of practice

Theoretical framework was created by reading and examining an existing bibliography of books and other relevant materials. Many of the researched writings discussed agile requirements management, agile software development, agile product management, and agile project management. These books were studied continuously during the course of the thesis project. Several articles and relevant blogs across the internet were also utilized relating to the subject. In addition, a number of recorded agile seminar videos were viewed while gathering and gaining material and knowledge for the thesis work. Basware's internal documentation about requirements management process was also investigated, discussed, and studied in the process.

The first part of the theoretical framework, chapter 4 "Requirements management practices and product management", discusses requirements management process and practices mainly from the product management point of view. Chapter 5 discusses hierarchical requirements structure and the new requirements hierarchy that was created and had been taken partially into use at Basware during this thesis project. Emphasis in Chapter 5 is on the user story requirement element. The last part of the theoretical framework, chapter 6, focuses on agile product management community practices and incorporates several agile requirements management best practices and conversation pieces. Various stories about ID project teams and Pilot project teams actions in practice are embedded into the theoretical framework creating a dialog between theory and practice. The next chapter, chapter 3, compares traditional software development with agile software development and presents briefly the Scrum methodology.

3 Product development approaches

Theory supports actions in practice. Agile methodology utilization in software development is an adaption process. Supporting processes, such as requirements management, must also change during the adaption.

The change from traditional software development model to agile model is a challenging and long exercise for an organization. Scrum methodology provides an agile framework for software product development, but the adaption is the key.

3.1 Waterfall vs. Agile methods

The waterfall model is suitable to use if the contract of the work and requirements and acceptance criteria is complete, correct and unambiguous. Work must be able to complete within constrains and where no change is expected in the requirements or design. The problem and solution is well understood, clearly defined and the likelihood of making mistakes in the requirements or design is low. But, software is too complicated and it is very rare for all these to be true. Something better is required. (Evans 2004, 199.)

According to Leffingwell (2011, 16), the fundamental difference between traditional and agile worlds culminates in the following two agile manifesto statements:

- Satisfy the customer through early and continuous delivery of valuable software
- Welcome even late requirement changes, agility harnesses change for the customer's competitive advantage.

As Figure 4 illustrates, in waterfall/traditional model, requirements are locked but resources and date dimensions are estimated whereas in agile model requirement dimension is estimated but resources and date are fixed:

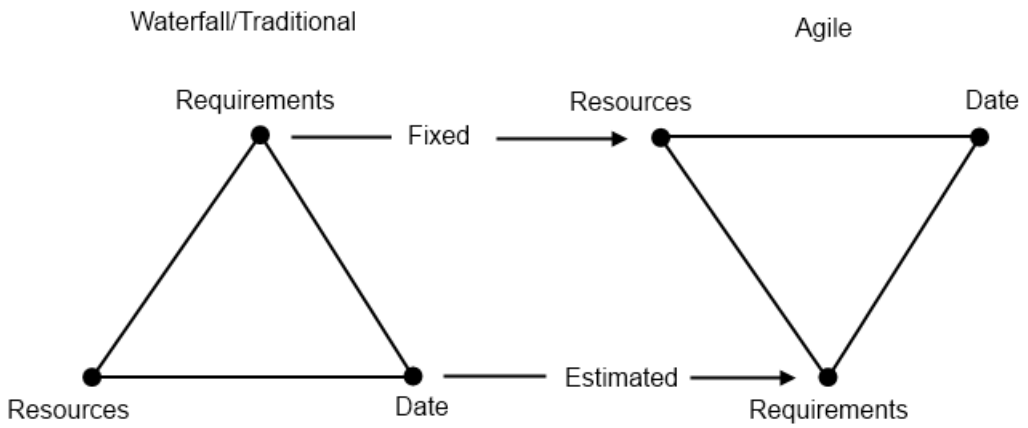


Figure 4. Agile locks the date and resources and varies the scope (Leffingwell 2011, 17)

Instead of building detailed software requirement specifications for months, more flexible approach to requirements management is taken with agile methods. Requirements management is characterized more as temporal, interactive, and just-in-time by nature (Leffingwell 2012, 18.) "Another difference between traditional approaches and agile methods is that in traditional approaches the customer is mainly involved during the early phase of the project while agile methods involve the customer throughout the whole development process." (Paetsch & Eberlein & Maurer 2003.)

According to the 2011 CHAOS report from the Standish Group, agile projects are successful three times more often than non-agile projects. The agile process has proved to be remedy for software development project failure. Software applications developed through the agile process have proven to have a much lower percentage of time and cost overruns. The Standish Group defines project success as on time, on budget, and with all planned features. It is unknown how many projects are in Standish Group's database but the results are from projects conducted from 2002 through 2010. The following figure 5 shows the results. (Cohn 2012b.)

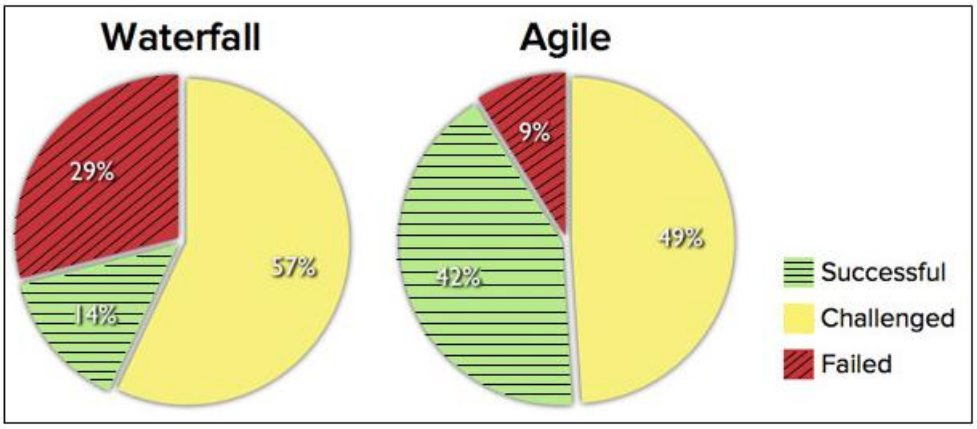


Figure 5. Waterfall and agile success-failure rates (Cohn 2012b)

Agile development accelerates the delivery of initial business value. Through a process of continuous planning and feedback, it is possible to ensure that value is continuously being maximized throughout the development process. Teams are able to align the delivered software with desired business needs, easily adapting to changing requirements throughout the process. By measuring and evaluating status based on working, testing software, more accurate visibility into the actual progress of projects becomes available. As a result of following an agile process, implemented software system better addresses business and customer needs. (VersionOne 2013.)

In iterative processes, in Scrum for example, a purposeful move away from the traditional big and upfront design can be identified. The fundamental idea is to more quickly discover the "real user requirements" in early iterations. This substantially reduces the overall risk profile of a project. (Leffingwell 2011, 11).

3.2 Scrum methodology

Scrum is a way for teams to work together to develop a product. Product development occurs in small pieces with each piece building upon those previously created. Building products one small piece at a time encourages creativity and enables teams to respond to feedback and change, to build exactly and only what is needed. Scrum is a simple framework for effective team collaboration on complex projects. It provides a small set of rules that create just enough structure for teams to be able to focus their innovation on solving what might otherwise be seen as an insurmountable challenge. Scrum sup-

ports the need to be human at work: to belong, to learn, to do, to create and be creative, to grow, to improve, and to interact with other people. In other words, Scrum leverages the innate traits and characteristics in people to allow them to do great things together. (Scrum.Org. 2009.)

Scrum.Org (2009) represents the fundamental simple scrum process and its three primary roles:

1. Product owners determine what needs to be built in the next sprint.
2. Development teams build what is needed in sprint, and then demonstrate what they have built. Based on this demonstration, the product owner determines what to build next.
3. Scrum masters ensure this process happens as smoothly as possible, and continually help improve the process, the team and the product being created.

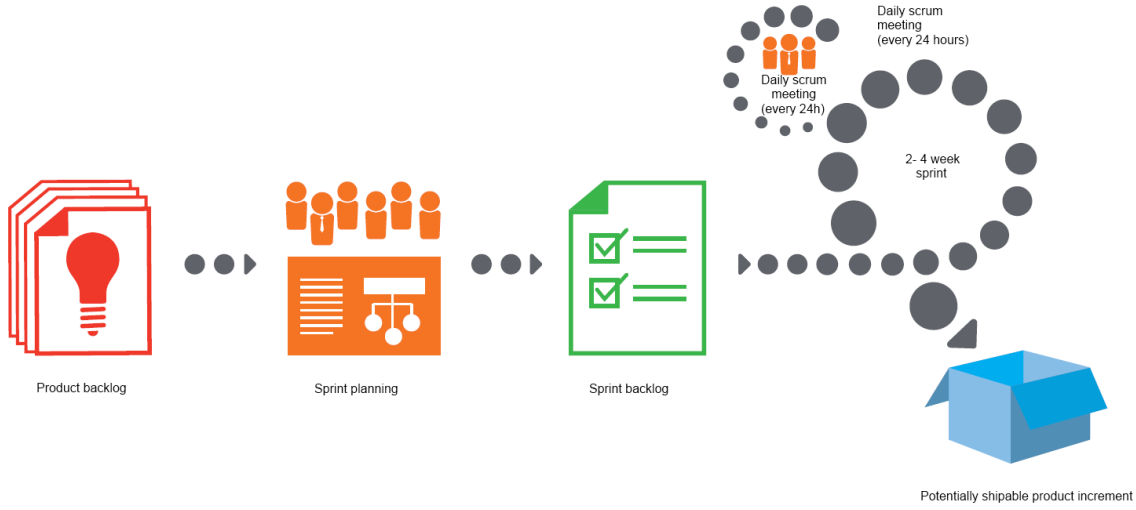


Figure 6. The scrum framework (Scrum Alliance 2013)

As visualized in figure 6, Scrum Alliance (2013) gives a more detailed description of the Scrum phases. Product owner is responsible to create a prioritized list called a product backlog. During sprint planning, the team pulls items from the top of that wish list, a sprint backlog, and decides how to implement those pieces. The team has a certain

amount of time, a sprint (usually two to four weeks), to complete its work, but it meets every day to assess its progress in a daily scrum meeting.

Along the way, the scrum master keeps the team focused on its goal. At the end of the sprint, the work should be potentially shippable: ready to hand to a customer, put on a store shelf, or show to a stakeholder. The sprint always ends with a sprint review and retrospective meeting. As the next sprint begins, the team chooses another set of items of the product backlog and begins working again. (Scrum Alliance 2013.)

Basware utilizes scrum methodology in all of the Alusta program projects. Product owner role is missing from Basware's adapted model. Product owner role is mainly played by product managers, but system analysts, architects, and project managers can also be recognized as product owners in some of the projects.

4 Requirements management practices and product management

The requirements management process manages all requirements received or generated by the project, including both functional and non-functional requirements as well as requirements levied on the project by the organization. The project should take actions to ensure that accepted and approved requirements are managed to support the broader planning and execution needs of the project. Often, the project usually receives requirements from an approved requirements provider and these should be reviewed with the requirements provider to resolve issues and prevent misunderstanding before requirements are included into further plans. Once the provider and the receiver reach a common understanding, commitment to incorporating and applying the requirements will be achieved by project participants. (Software Engineering Institute 2010, 341.)

In agile development environments, requirements are communicated and traced by utilizing mechanisms such as product backlogs and story cards. Commitments to requirements are either authorized by the team leader or collectively by the team. Work

assignments are regularly adjusted based on progress and as an improved understanding of the requirements and solution emerge. Traceability and consistency across requirements and work products is addressed through the mechanisms already mentioned as well as during start-of-iteration or end-of-iteration activities, of demo days which are part of the scrum model process. (Software Engineering Institute 2010, 342.)

According to H2 2012 ID project roadmap, the project team created a short term plan for system analysts. Few usefull practices were identified which were not in use among most of the teams working for the Alusta program. At that time, preparation for an iteration was suffering from discontinuous communication and poor co-operation. In particular, collaboration between architects and system analysts was identified as beeing poor. It was decided by the ID project team that it would present handfull of best practices to system analysts provide them with the responsibility to implement these new practices into project practices.

Based on the questionnaire results, noticeable in Attachment 6, it became apparent that requirements review practice was absent partially from Alusta projects practices at which point the ID project team decided to include it as part of their short term improvement. Change in project practices was to pair a system analyst with an application architect on the working day following each sprint planning meeting. An hour meeting was suggested where these two team members would kick off the specification work for the next release and to make sure that they shared the same understanding of the content. A preliminary requirements review for the next sprint content was included as part of this same meeting.

In larger enterprises, groups of agile teams work together to support building up larger and more complex functionality into complete products, features, architectural components, subsystems, and so on. Stakeholders need to understand the impact of requirements practices on departmental and organizational activities – not just at the individual project or team level but all the way to the enterprise portfolio – because that is where the new projects are formed. (Leffingwell 2011, 27, 33.)

The second practice improvement introduced was a recurring weekly project management meeting. Participants would consist of a project manager, a product manager, system analysts, application architects, and chief architect. Led by the project manager, the meeting was intended to focus on teams specification work assesments, specification dependencies between other teams specifications, and the overall requirements readiness status for the next iteration. Project manager would go through each teams status by interviewing system analyst and architect. This meeting also proved to be great practice to share an overall understanding of the current release situation between different project teams and their members.

One of the most significant challenges in system development is misunderstood requirements. In order to avoid misunderstandings, requirements must created in an unambiguous and testable way. Equally, it is important to ensure the originating owner of the requirement understands and is aligned with the written requirement before it is given on to the development. Requirements should be written in a way that parties at either end of the development line have similar understanding of what is needed. Requirements must have the same meaning to the stakeholder who orginated it as to the developer who will build it." (Robertson & Robertson 2012, 45-46.)

The ID project team predicted, that the emergence of a user story requirement element at Basware will most probably improve common understanding of customers needs between different stakeholders. The purpose of the user story element is presented and discussed later in this study.

4.1 Introducing the Pilot project team

In the beginning of 2013 an unexpected input was provided to ID project by Alusta purchase project which was simultaneously seeking out solutions for weak existing requirements management practices. Purchase project, later called Pilot project, had previously encountered situation where the Alusta program manager had asked the project to define its own targets which could be reached in a short period of time. The Pilot

project defined its own targets, however was unable to reach those. This outcome quickly turned the focus on requirements management process inside the Pilot project.

The Pilot project started to sketch hierarchical structure for requirements. At that time, requirements were created and managed mainly by system analysts in the Pilot project. As the amount of detailed specifications was increasing all the time, backlogs became more and more challenging to maintain and follow. A hierarchical requirements approach was needed and a draft of the new requirements hierarchy, visualized in Figure 7, was presented to the ID project team:

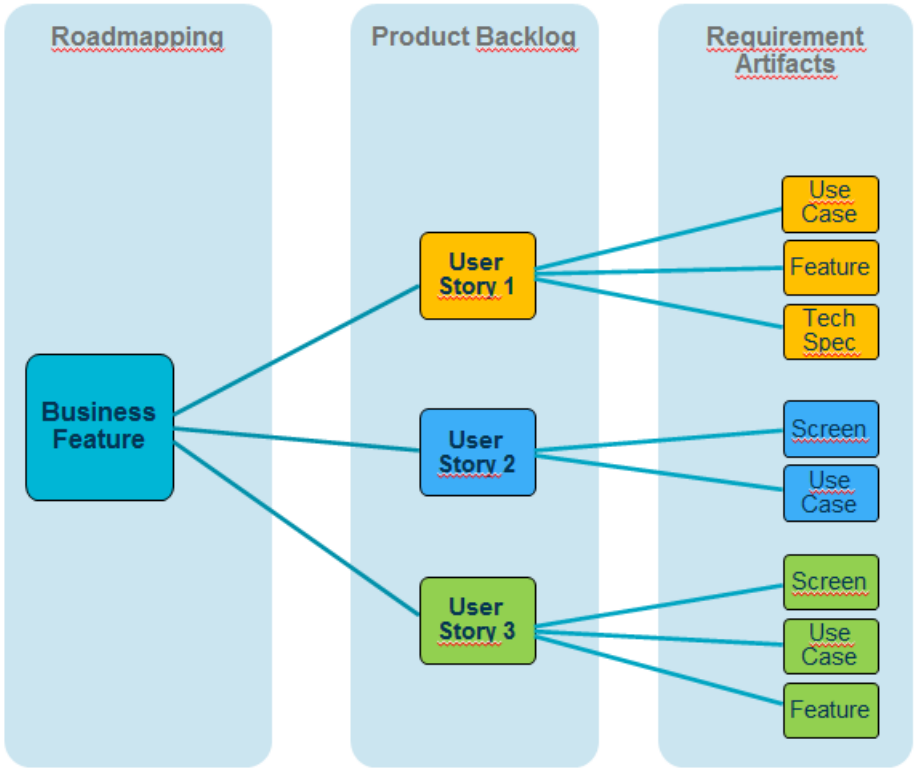


Figure 7. Draft version of requirements hierarchy provided be pilot project

The Pilot project management team consisted of a product manager, four system analysts, chief architect, four application architects, and a project manager. As the product manager of the Pilot project also happened to be the mentor of this thesis work, requirements management changes in practices were quickly applied to Pilot projects routines. This setup was ideal for the ID project as well, independent Pilot project brought in valuable results from the field and was ready to implement and develop its working practices and methods.

4.2 The role of product management

One of the main objectives of agile method, is for everyone to know what to expect and when to expect it, after repeated iterations and releases. Stakeholders, such as product management, see progress frequently and substantially, working software. This enables instant feedback mechanism. (Martin R. 2012, 22.)

Management rarely has direct authority over the people working on the product, and this must be fully understood that leveraging the talents of a large pool of experts is far more effective than trying to control the work itself. The development team must be lead instead of trying to manage it. (Poppendieck & Poppendieck 2003, 112.)

The Pilot project started to arrange monthly workshops lasting for half a day where a product manager went through preselected business requirements by explaining them with real life examples. By sharing this knowledge, system analysts expanded the understanding of the business areas and needs. One day was also dedicated solely for the release kick off meeting where product manager introduced the new release theme and related business features.

Leffingwell (2011, 228) reminds that workshops are just one tool for requirements discovery. Brainstorming, interviews and questionnaires, and for example competetive analysis are also techniques that can help team to discover requirements. As there usually is no "one-size-fits-all" approach, a combination of these techniques are required and can be used for any particular circumstance.

4.3 Product management responsibilities

Management, chief architects, analysts and experienced developers are experts in the domain and the technology; they understand both the customers and developers. They understand the system's constraints, interactions, unstated requirements, and exception conditions. They look at the system from a fairly high level of abstraction, but can drill down to the complex parts and detail that both developers and customers must cope

with. They have the wisdom to guide market trade-offs in product development and business trade-offs in internal development. (Poppendieck & Poppendieck 2003, 113.)

Maciaszek (2005, 59-62) categorizes management responsibilities into three main areas:

1. identifying, classifying, organizing, and documenting the requirements.
2. changing requirements
3. creating traceability for requirements.

As system may consist of hundreds or thousands of requirements, these must be able to manage effectively. Maintenance of multiple requirements versions, links between different specifications enables support for requirements change management and traceability. Ability for creating hierarchical structure is also recommended for requirements. And as requirements tend to change, unmanaged change is a kick in the teeth. A requirement may change, be removed, or a new requirement may added at any phase of the development lifecycle. Requirements traceability is about maintaining traceability relationships to track changes from/to a requirement throughout the development lifecycle. Without tool support, requirements management is doomed. (Maciaszek 2005, 59-62).

Each target project of the thesis uses cloud based application, Accept 360, for requirements management. According to Accept Software (2012), the application provides solutions that enable to collect, organize, and manage product requirements, ideas, product strategies and portfolios, and agile execution information all in one place. It allows the entire product organization to view all product planning and execution data in context with business goals, priorities, and development implications, while keeping everything in sync as conditions change.

4.4 Identifying needs and requirements

Business analysis is the activity of determining and specifying customer requirements. The task of the requirements determination phase is to determine, analyse, and negotiate requirements with customers. Analysis phase includes negotiations between devel-

opers and customers. This step is necessary to eliminate contradicting and overlapping requirements, and also to conform to the project budget and deadline. Specification work allows the proposed solution to be viewed from many angles so that specific aspects of the solution are emphasized and analysed. Consistency and completeness of requirements are also carefully checked. (Maziascek 2005, 22-23).

"The majority of people who collect and manage requirements don't follow a specific methodology, since it's often considered less important than the follow-on development or implementation activities. What's more, most people don't realize that requirements management is a collaborative activity" (Lahanas 2013).

Pilot project team felt that there were too many items in the backlog and it tried to serve too many stakeholders. This made it hard to prioritize but also hard to estimate items. Backlog items didn't translate nicely into business features or into end to end use cases which could be utilized for example by Alusta system testing teams. It was difficult to see what features were completed and which not.

According to Pilot project team, it was also challenging to schedule the work and make sure that features will be finished. There was simply too much variation on how different items were used, how different sub-backlogs were managed and what was the level of detailed specifications. It was very hard to find and see relations between items. Several backlog "stirring" exercises were needed during the existing project in order to manage team and release backlogs, update work estimates etcetera. It was very useful work but with more structured overall approach a lot of time could have been saved and used for other important activities. In addition to team and release backlogs, the product backlog started to swell unnoticed as requirements kept emerging from several different sources.

> As the project matures and requirements are derived, all activities or disciplines will receive requirements. To avoid requirements creep, criteria are established to designate appropriate channels or official sources from which to receive requirements. Those who receive requirements conduct analyses of them with the provider to ensure that a compatible, shared understanding is reached on the meaning of requirements. The

result of these analyses and dialogs is a set of approved requirements. (Software Engineering Institute 2010, 343.)

4.5 Analysing requirements

Product does not change the real nature of the work; it just changes the way it is done. Before one can design an optimal product, one must understand the work it supports. Most importantly, one must understand client's desired outcome of the work. In the first phase all the technicalities and details of the business case can be forgotten and instead one should look outside the organization to see what kind of response is needed or wanted by the organization's customers and suppliers. (Robertson & Robertson 2006, 78-79). Requirements determination is about social, communication, and managerial skills. If not done thoroughly the consequences can be serious (Maciaszek 2005, 48).

There are often gaps in time between when requirements are defined and when development starts it work. There are many reasons for this to happen, but it's important to understand that this occurs on a regular basis. The larger the gap is in time between definition of requirements and development, the more the risk that occurs with developing things right. The longer the gap is, the higher is the risk of losing team members, knowledge, and overall team availability. Decomposition is a way to pick up where things left off, by using the product backlog to communicate and share requirements. (Moccia 2012.)

It got evident for the Pilot project team that product backlog view with all its requirement types was not necessary to be shown for example for product manager. In order to keep the complexity of the requirements manageable for the project staff, it was necessary to find way to filter the requirements view according to the task. With a help of requirements management tool owner, the product backlog filtering for example by requirement type was taken into use. It was existing functionality but wasn't used before. This allowed management team to easily view the content on a business feature level which made it easier to effectively prioritize work and start better understanding the big picture.

One way, probably the most important one, of ensuring incremental business value along with the productivity is to prioritize the requirements effectively and give the best possible understanding of this requirements prioritization to the team. Product backlog grooming helps to add business value. (Pund 2013).

Requirements influence the analysis, design, implementation and test phases directly and indirectly. The quality of a requirement or of a requirements document has an impact on the progress of the project and therefore on its success. (Pohl & Rupp 2011, 33.) One could say that software without documentation is a disaster and code should not be the medium for communicating the rationale and structure of a system. Human-readable documents that describe the system and the rationale for their design decisions must be produced. (Martin 2012, 5.) Pohl & Rupp (2011, 10) further continues that the goal is that requirements should be documented as completely as possible in good quality and to identify and resolve problems as early as possible.

4.6 Changing requirements

The movement to more "agile" and "leaner" software development methodologies, including lighter weight but still safe and effective treatment of application requirements, has been one of the most significant factors affecting the industry. (Leffingwell 2011, 4). The number of methods – including for example Scrum, Extreme Programming (XP), Agile RUP, Kanban, Lean, Crystal Methods, and so on-speaks to the industry's thirst and constant drive for more effective and lighter-weight processes. (Leffingwell 2011, 12.)

Where Scrum treats requirements like a prioritized stack called a product backlog, some other methdos takes it one step further to recognize that not only do you implement requirements as part of your daily job but you also do non-requirement related work such as take training, go on vacation, review products of other teams, address defects and so on. With this approach software development team has a stack of prioritized and estimated work items, including requirements, which need to be

addressed. Stakeholders are responsible for prioritizing the requirements whereas developers are responsible for estimating. (Ambysoft Inc. 2005-2012a.)

Because requirements change frequently you need a streamlined, flexible approach to requirements change management. Agilists want to develop software which is both high-quality and high-value, and the easiest way to develop high-value software is to implement the highest priority requirements first. This enables them to maximize stakeholder ROI. In short, agilists strive to truly manage change, not to prevent it. (Ambysoft Inc. 2005-2012a.)

Requirements change process is unclear in more than half of the target projects and this is seen in the Attachment 4. Other finding also supported this result: bugs are not mapped into existing requirements in more than half of the projects. This strengthened ID project team's view on the change management process but at the same time it was too early to start fixing change management process before requirements management fundamentals were clarified from top to bottom.

Often requirements tend to change for different reasons. As needs change and as work proceeds, it is essential to manage these additions and changes efficiently and effectively. Changes may have to be made to existing original requirements. Impact of changes should be analyzed effectively and it is necessary that the source of each requirement is known and the rationale for the change is documented. All requirements and requirements changes should be documented that are given to or generated by the project. It is desirable to maintain a requirements change history, including the rationale for changes, because this helps to track requirements volatility. Requirements changes that affect the product architecture can affect many stakeholders and requirements and change data should always be available to the project. (Software Engineering Institute 2010, 345.)

4.7 Creating baseline for solid and healthy product life cycle

"Agile development methodologies (such as XP, Scrum, and ASD) promise higher customer satisfaction, lower defect rates, faster development times and a solution to

rapidly changing requirements" (Boehm & Turner 2004). Requirements traceability supports change management. Therefore an important quality criterion is traceability of relationships between different requirements. (Pohl & Rupp 2011, 43.)

According to Boehm & Turner (2004), "Agility is the counterpart of discipline. Where discipline ingrains and strengthens, agility releases and invents. It allows the athlete to make the unexpected play, musicians to improvise and ornament, craftsmen to evolve their style, and engineers to adjust to changing technology and needs. Agility applies memory and history to adjust to new environments, react and adapt, take advantage of unexpected opportunities, and update the experience base for the future."

As Pilot project had started requirements management work for the coming release it was realized that before starting creating documentation for the coming next release, each and every requirement document should be updated before proceeding any further. State and flow diagrams, use cases and other requirements had to be updated, it was the starting point for the continuous release cycle after all.

Pohl & Rupp (2011, 136) reminds that during requirements management, traceability of requirements is created, organized, and maintained. Information about cross references and dependencies between requirements are created. Versioning and changes of the requirements helps management to keep requirement data up to date. Documents should be available over the whole life cycle of the product. Requirements are prioritized constantly, during different activities according to different criteria.

Cockburn (2007, 56) believes that software development can be fit in the proper engineering tradition by building on the foundations of craft. It is a cooperative game of invention and communication. Some need people management skills, others need mathematical skills, and so on. Requirements elicitation is one important element in software development craft together with fox example project management, UI design, programming, testing, and communication.

At Basware roadmaps had traditionally been communicated with promised delivery dates and fixed content. In practice, Pilot project had problems meeting both the schedule and the content. Main driver for changing roadmapping approach was to improve related customer satisfaction by eliminating disappointments regarding missed schedules and features.

One of the issues that had to be addressed was the fact the with a new product the development priorities have to be decided based on the acute customer needs. With traditional approach pilot project would have most probably ended up in a situation that roadmap promises would have possibly forced project to implement promised features and not to focus on other features that would have had a bigger demand and better market potential. Changed approach gave pilot project flexibility to react to rising customer requirements from the active projects rather than following outdated plans which had been promised to individual customers by contracts.

Third aspect was linking product strategy, product features and actual product development work tighter together. Changed backlog management practices made it possible to steer development more efficiently on a daily basis but also helped project management to link day to day activities to business features, product themes and there into the roadmap. Pilot project team did not only started prioritizing but also sequencing lower priority tasks that must be completed first because there was a dependency from higher priority elemement.

5 Managing functional requirements with new method

It is faster and more cost efficient to let features evolve over the life of the project than hallucinate about what the requirements are in the beginning and then build them without constant customer or customer representative involvement. A strategy of building light features and a capability to easily and reasonably adapt to change can be very profitable. Everyone and everything should focus on key vision and goals and trade-off decisions must be welcome. Product scope should not be held hostage to a

plan developed when product and project knowledge was still in its infancy. (High-smith 2004, 158-159.)

In the iterative model, lighter-weight documents and models are applied, such as vision documents, business feature descriptions, and so on. These are used to initially define what is to be built. Based on these initial understandings, the iterative process itself is then applied to more quickly discover the the "real user requirements" in early iterations, thus substantially reducing the overall risk profile of the project. Once better defined in early iterations, these requirements are then implemented in a fairly robust but mostly traditional build-out of code, tests, and so on, to implement the requirements and provide assurances that the system conformed to the agreed-to behaviours. (Leffingwell 2011, 11.)

5.1 Requirements hierarchy

"Most of the time when people refer to a requirement, they are thinking of a short description of a single feature the product is required to have". (Garret 2002, 69.) On the other hand, requirement can be something that people don't know they want. This leads to solution where requirements generation often is a matter of finding ways to remove impediments. If the scope of the project allows, very detailed level requiments should be used. Requirements shoud always be up-to-date, no matter the phase the project is going (Garret 2002, 70-71.)

According to Leffingwell (2011, 87) requirements are all forms of expressing user need and implied benefit. Different level of abstractions leads to hierarchical model which reduces the level of too-early specificity and decreases the overhead of managing re-quirements for larger systems. Requirements can be categorized into three different levels where epics form the highest level requirement group followed by features at the middle level and finally user stories at the bottom:

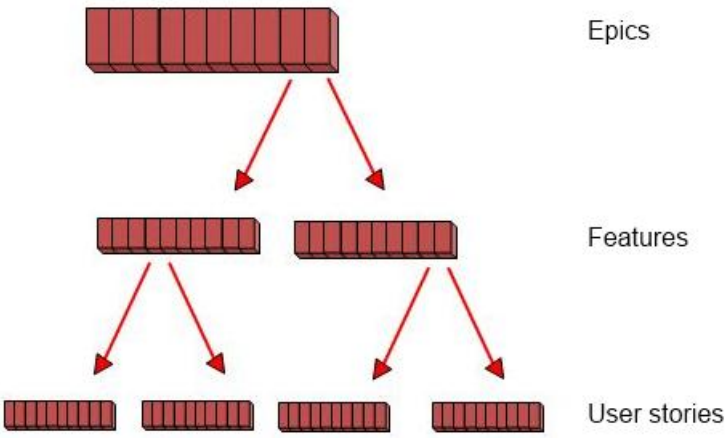


Figure 8. Hierachical view of requirements (Leffingwell 2011, 87)

Leffingwells requirements information model, Figure 8, consists of epics, features, user stories, and backlog item objects. Basware's approach follows this model, epic level is known as vision, features are called business features, and backlog item objects are recognized as artifacts.

Cardinalities between objects are marked to the figure 9.

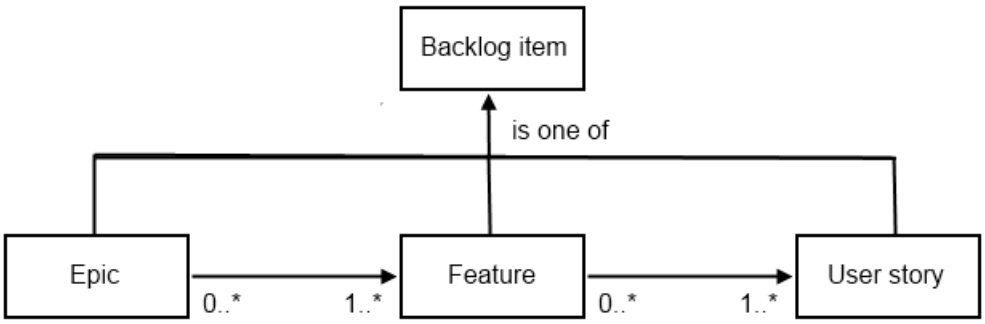


Figure 9. Epics, features, and user stories in the requirements information model (Leffingwell 2011, 86)

Leffingwell (2011, 251-252) highlights the difference between traditional and agile development. Investment in up-front requirements analysis is reduced in agile. Instead of wasting time with traditional marketing, product and software requirements documents agile enterprises take a leaner approach better suited to the last-responsible-moment, delayed decision-making, and artefact-light development practices. The idea is to prevent overinvestment is things that are unlikely to understand very well anyway and to

prevent the too-early binding of resources to a set of fixed commitments that are likely to haunt management later. Now, when traditional documents do no longer exist to specify the system behaviour, it is even more critical to communicate the vision.

5.2 Vision

Pichler (2010, 24) says that vision acts the overarching goal, galvanizing and guiding people. It is the product's reason for being. Vision should answer at least the following questions:

- Which needs will the product address and what values does it add?
- Which product attributes are critical for meeting the needs selected and therefore for the success of the product?
- What will the product roughly look like and do?
- In which areas is the product going to excel?

Leffingwell (2011, 40, 252) continues the list with questions:

- What problem does the solution solve?
- For whom does it provide it?
- What features and benefits does it provide?
- What non-functional values does it provide or support?

Pichler (2010, 25-26, 40) says that vision should communicate the essence of the product and describe a goal that provides direction but is broad enough to facilitate creativity. Every stakeholder must share the vision that gives direction towards the common goal. When product matures and updates are released, the visioning effort decreases but new versions still need goals. Product roadmap allows everyone to capture the goals of upcoming versions. Visioning should form a part of creating and maintaining the product road map.

Hightsmith (2004, 66) reminds that vision and desired end result communication should be on going task of management. Products evolve, there are changes in resources and during the project the vision becomes blurred. Vision is not a one-time activity. Management also need to communicate a sense of what the new important areas are and why they are high priority.

Product vision creation is an important step in taking a product to launch. When there is no vision, most often individual features requested by customers are incorporated into the product with no consideration of the link between them. To minimize potential loss or damage form an inaccurate forecast, a narrow set of needs should be selected and focus should be turned to the next release increment. Visioning work should be kept to a minimum to prevent so called analysis paralysis. Get the product out fast and swiftly adapt it to the actual market response. "We know the best what is good for our customers" -attitude must be avoided. This easily leads into a product that nobody wants because all the innovation related work has been done internally. Also a product release with abundance of functionality requires usually a lot of effort, time and money. This so called big-bang endeavour makes it difficult to evolve the product on customer feedback as so much functionality is predetermined. (Pichler 2010, 43-45.)

Referring to Attachment 5, majority of respondents stated that product line vision is clear in target projects. This created belief in ID project team: vision could be included to the requirements hierarchy without paying too much attention to this particular element and implementation work. The focus was decided to be kept on business features and especially on user stories.

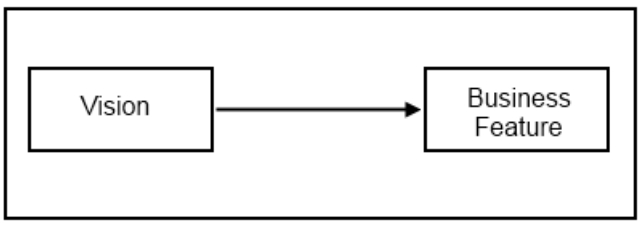


Figure 10. Vision and business feature

It was identified by the ID project team that the role of the vision should be communicated clearly because business features are always led from the vision as seen in the Figure 10 above.

5.3 Business features

Leffingwell (2011, 75) says that business features are led from vision. Features describe what new things the system will offer to users and the benefits the user will gain. Robertson & Robertson (2006, 75) state that business features point out which things belong together and delivers harmonious partitions with minimal interfaces between different pieces. Partitioning, in turn, leads to smaller pieces of work that help the investigation of detailed requirements. The fewer the dependencies that exist between pieces, the more the analysts can investigate the details about one piece without needing to know anything about others.

The ID project team renewed business feature element template in Accept 360 and included example text from one of the Pilot project business feature:

<Business feature describes the business need from the customer's point of view, gives high level overall vision of functionality and perspective of the related customer needs. Business need does NOT describe the functionality on UI or feature level>

Example text from Pilot project's existing business feature called "item comparison", was added to the business feature element type:

Product comparison is a basic tool that enables users to see several items and their details at the same time. Comparison helps users to make a selection between different items and find out differences between different items. Using comparison should be easy and functionality should be easily available to user. User should be able to add and delete items from the comparison as he/she progresses with his/her search and finds more items to review. Comparison must be easily turned on and turned off.

Leffingwell's (2011, 260) model links user story element to feature element, as seen in Figure 11 below.

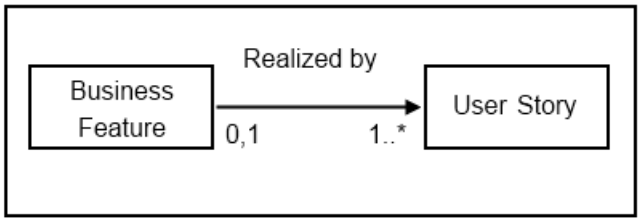


Figure 11. Feature relationship to Story (Leffinwell 2011, 260)

5.4 User stories

Something fascinating happens when requirements are put in the first person. Obviously by saying "As a such-and-such, I want …" person's mind goes instantly to imagining he or she is a such-and-such. As for the magic, Paul McCartney was interviewed and asked about why the Beatles songs were so amazingly popular. One of his responses was that their songs were among the first to use a lot of pronouns: She Loves You, I Wanna Hold Your Hand, I Saw Her Standing There etc. His point was that these helped people more closely identify with the songs. (Cohn 2008.)

User stories should be short and story should represent a concept. Each story must provide something valuable to a customer. Stories require communication between customer and development team, customer provides stories and team should put most of their attention on technical decisions (Beck & Fowler 2001, 46-47). A user story is a reminder to have a conversation with project stakeholders. User stories capture for example high-level requirements, including behavioural requirements, business rules, constraints, and technical requirements. (Ambysoft Inc. 2001-2012b.)

It is very important, according to Cockburn (2007, 286), to present user stories with operations context and to be able to express and communicate related user stories and user stories that surround the ones that are going to be implemented. It should be possible to integrate even hundreds of tiny user stories into a coherent whole.

5.4.1 Form of a user story

Leffingwell (2011, 103) opens the form for a user story construct as followed:

<user role>	denotes who is performing the action or who is getting the value from the activity.
<goal>	represents the action to be performed
<business value/reason>	denotes the value achieved by the activity

User stories help expressing business value continuously. Idea is to avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution. Stories should also avoid the appearance of false completeness and clarity. It is important to get to small enough chunks that invite negotiation and movement in the backlog. Technical functions can and should be left to architects, developers, and testers. (Rally Software 2013.)

ID project decided to add a new template for user story elements in Accept 360 application containing the syntax and an example user story as followed:

As <user role>, I <want/need/can etc> <goal> so that <business value/reason>

** <As a Needer I want to see price differences between two catalog items, which helps me to make the purchase decision.>*

As user stories are driven by business features they are often very large in the first face. They present a vague concept of something that needs to be done. This also means that stories which are way too big to be implemented within an iteration, those must be broke down into smaller stories. (Leffingwell 2011, 111.)

5.4.2 Splitting user stories

Lawrence (2009) introduces patterns for user story splitting. In a workflow related story, the biggest value often comes from the beginning and end. The middle steps add incremental value, but don't stand alone. It can work well to build the simple end-to-end case first and then add the middle steps and special cases. Business rule variations are often identified with a story that has a few equally complex stories hidden within it that accomplish the same thing using different business rules.

Lawrence (2009) also says that sometimes a story can be split into several parts where most of the effort will go towards implementing the first one. For example, this credit card processing story: "As a user, I can pay for my flight with VISA, MasterCard, Diners Club." This could be split into three stories, one for each card type. But the credit card processing infrastructure will be built to the support the first story; adding more card types will be relatively trivial. Estimation can be done for the first story larger than the other three, but then it is crucial to remember to change estimates if priorities changes later. Instead, it is wise to defer the decision about which card type gets implemented first. The two new stories still aren't independent, but the dependency is much clearer than it would be with a story for each card type.

…I can pay with one credit card type (of VISA, MasterCard, Diners Club).
…I can pay with all three credit card types (VISA, MasterCard, Diners Club) (given one card type already implemented).

Sometimes when having a discussing about a story, the story seems to get larger and larger, it is good to stop and ask, "What's the simplest version of this?" Simple version should be captured as its own story. It is also recommended to define some acceptance criteria on the spot to keep it simple and then break out all the variations and complexities into their own stories. Complexity in a story can come from handling variations in data. Usually it is important to find an answer to question "What's the 'good enough' way to model the core functionality so that other high-value features can be built now?" Sometimes data variations are known up-front like in localization cases. (Lawrence 2009.)

Complexity sometimes is in the user interface rather than in the functionality itself. In that case, a story can be chopped to build it with the simplest possible UI and then build the more usable or fancier UI. These, of course, aren't independent—the second story effectively is the original story if it is done first—but it still can be a useful split. Sometimes, a large part of the effort is in making a feature fast—the initial implementation isn't all that hard. A lot can be learnt from the slow implementation and it still

has some value to a user who wouldn't otherwise be able to do the action in the story. In this case, the story can have two categories: "make it work" and "make it fast": The word "manage" in a user story is a giveaway that the story covers multiple operations. This offers a natural way to split the story. (Lawrence 2009.)

Lawrence (2009) takes investigation stories to the picture by pointing out that a story may be large not because it's necessarily complex, but because the implementation is poorly understood. In this case, no amount of talking about the business part of the story will allow one to break it up. A time-boxed spike should be done first to resolve uncertainty around the implementation. Then it could be implemented or even a better idea of how to break it up might be born. In the "investigate" story, the acceptance criteria should be questions what needs to be answered. Just enough investigation should be done to answer the questions because it's easy to get carried away doing research. The spike split is last because it should be the last resort.

When the Pilot project started to create user stories during H1 2013, it was quickly realized that stories needed lot of attention. Because of highly configurable application, user stories were often split into configuration and utilization pieces. One of the biggest challenges was to connect user stories to existing functionalities and especially to functional areas adjacent to new stories.

5.4.3 Acceptance criteria

Good acceptance criteria will help agile project from "It works as coded" to "It works as intended". Acceptance criteria provide a way to clearly demonstrate if the development team has made the user story come true. They are a set of statements, each with a clear pass or fail result, that specify both functional and non-functional requirements applicable at the current stage of project integration. These requirements represent conditions of satisfaction and there is no partial acceptance: either a criterion is met or it is not. (Jackson 2013.)

The ID project team modified the user story template by adding a section for acceptance criteria:

* The product owner's and architects conditions of satisfaction
* Functional:
    - <Verify that an email confirmation is sent.>
    - <Verify that the hotel is notified of any cancellation.>
* Non-functional:
    - <Searched items must be returned in 4 seconds>
    - <One click approval>
* Level/size of user story: t-shirt sizing (scale: S,M,L,XL)

The ID Project team noticed that it is important to include non-functional requirements stronger to programs everyday work. Non-functional requirement could be part of the user stories or handled separately in detailed technical specifications. Regardless of the location, person responsible for creating non-functional specifications should think and find out at least the need of the fastness of different actions.

It is also important to know user's profile and understand user's expectations. Management or architects should be able to tell how frequently for instance a use case is executed. It is valuable to know if the functionality is used only occasionally and so forth it is not a performance critical one. On the other hand users might use functionality hundreds of times per day and they expect windows client type of performance; "Anything over 3 sec is not acceptable, less than 1 second would be good". From the performance point of view it is also good to understand priorities; "Opening should be like on other pages and max 3 sec is ok, additional actions may take longer".

In addition to acceptance criteria, additional notes and assumptions can also be kept with a user story. Alternative flows, acceptance boundaries, and other clarifications can be captured along with the story. Often many of these are turned into acceptance test cases, functional test cases. (Leffingwell 2011, 104.)

The problem with missing acceptance criteria is that assumptions often differs, some might assume that criteria are too obvious to mention and they may be interpreted dif-

ferently by other people (Evans 2004, 174). Acceptance criteria must be expressed clearly, in simple language, just like the user story, without ambiguity as to what the expected outcome is. They must also be actionable: easily translated into one or more test cases. These criteria define the boundaries and parameters of a user story and determine when a story is completed and working as expected. They add certainty to what the team is building. (Jackson 2013.)

5.4.4 Same language, different stakeholders

Stakeholders must not be requested to learn another new notation. By using natural language, like user stories, any kind of requirement can be expressed. However, there is a danger that requirements become too ambiguous and requirements of different types and perspectives are being unintentionally mixed up during documentation. (Pohl & Rupp 2011, 35.)

Leffingwell (2011, 101) reminds that in agile development, effective communication with common language is the key. It is development's job to speak the language of the user, not the user's job to speak the language of developers. Cohn (2012a) points out the main purpose of a user story: story text itself is less important than the conversation stakeholders have. In general, focus should be shifted from writing to talking.

5.4.5 Benefits

Acceptance criteria as part of a user stories has several benefits. They help the team to think through how a feature or piece of functionality will work from the user's perspective. Acceptance criteria remove ambiguity from a requirement and they form the tests that will confirm that a feature or piece of functionality is working and complete. (Boost New Media 2010.)

According to Cockburn (2007, 284-285) user stories can be very useful in certain situations but can also make a weak strategy in others. Risk, in situations when systems serve diverse set of users in multicultural environments, exists. Using only user stories in wrong places often causes damage because of tunnel vision, lack of context, and lack

of completeness. Well written use cases, in addition to user stories, can solve these three problems.

Cockburn (2007, 288-289) presents Gerard Meszaros's (2004) findings about the symbiosis between user stories and use cases. As a user story may for example be a request for new functionality, or improved usability it is critical to understand that relative use case must not contain, in any event, data or UI design information. A user story must be small enough to be implemented in iteration. A use case may contain even dozens of user stories, but it is not the case that each step in use case is a user story. The first user story to be implemented should be the simplest path through the main success scenario and it should map to multiple use case steps. Extension conditions, such as alternative flows, in a use case can map to several user stories. Some steps describe for example business rules that are challenging to implement, either of which may need to be split into new user stories, implemented in different iterations.

The cardinality between user stories and artifacts at Basware is shown in Figure 12:

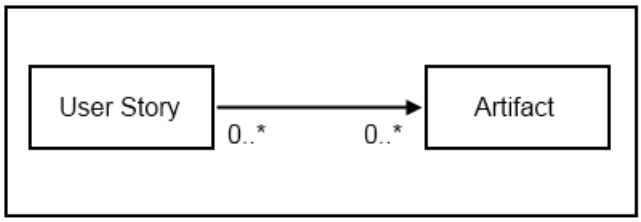


Figure 12. User story and artifact element relationship

5.5 Detailed specifications

The safest way to ensure that the users needs are met is to write down, what the users need and what a system would have to do to meet that need. These are the user requirements and the system specifications respectively (Alexander & Stevens 2002, 1-2). Pohl & Ripp (2011, 42.) advises that requirements document should be appropriately comprehensive and clearly structured in order to guarantee that it is readable by any stakeholder. Ambler (2009) continues by saying that documentation should be written as late as possible, avoiding speculative ideas that are likely to change in favour of sta-

ble information. Early investment in specifications will likely prove to be a waste due to the speculative nature of requirements definition.

Before the requirements management ID project initiated, detailed specifications, artifact elements in other words, had dominant role in the requirements management and specifications area. Only handful of development projects had linked artifacts to business features, user story element barely existed. And if user story was used, it sure wasn't linking business features and artifacts together. The general problem at Basware was that it was almost impossible to follow and understand statuses of current projects because of inconsistent usage of these different requirement elements and the missing linkage to any other entities of features. Although the life cycle status of an artifact element changes after development has been completed, nobody can really tell whether the functionality was ready to be released or not.

There are still 16 different types of requirement elements available in Accept 360 application but project team recommends usage for only 3 of those; use cases, graphical user interfaces and technical specifications. System analysts are mainly responsible for use cases. Screen elements are owned by the ux team and technical item contains mainly architects output and specification to help teams to build cohesive solutions effectively.

"If the description of the user story is too complex for natural language and if the business cannot afford to have the specification misunderstood, the team should augment the story with more precise specification method" (Leffingwell 2011). Requirements move from analysts to designers, and the design documents move from designers to developers, and the code moves from coders to testers, and so on. Each handoff of an artifact is fraught with opportunities for waste. The biggest waste of all in document handoffs is that documents don't contain all of the information that the next person in line needs to know. Great amounts of tacit knowledge remain with the creator of the document and never get handed off to the receiver. Moving artifacts from one group to another is a huge source of waste in software development. (Poppendieck M. & Poppendieck T 2003, 130).

Beck & Fowler (2001, 115) points out that bugs should be prioritized among all the other tasks in the release backlog. If bug is more than a day's effort it should be treated as a story. This leads to making tradeoffs between fixing defects and creating new features.

5.5.1 Use Case

Use case term was originally coined by Ivar Jacobson. His aim was to break the system into smaller units as he felt that object models were not scalable. To beat the complexity and largeness of systems, it is necessary to partition requirements into convenient chunks, and that chunks could be based on the user's view of the system. Jacobson presented the term actor to mean a user role or another system that lies outside the scope of the system. System in this usage is presumed to be an automated system. It is very important to be able to show the work's connections to the outside world with context diagram. Use case diagrams for example work well in illustrative purposes (Robertson & Robertson 2006, 69-70, 511) Cockburn (2007, 287) highlights that use case specification document should not contain any images or pictures in it.

Documentation should be done using templates that are easy to write, easy to read and easy to maintain. There is no need to determine all detailed requirements before coding. However, enough requirements should be collected so that programmers can code without having to make design choices at the same time. New requirements may pop up because of new business needs or because of something that came up while coding or testing, these should be incorporated back into the requirements first. Requirements document should be the most accurate and up-to-date artifact of the project. (Krisna 2007.)

Techopedia (2010-2013) explains that use cases define interactions between external actors and the system to reach particular goals. Actors are the type of users that interact with the system. Use cases capture functional requirements that specify the intended behaviour of the system. Use cases are usually initiated by a user to fulfil goals describing the activities and variants involved in reaching the goal. Use cases are typically

modeled using unified modeling language and are represented by ovals containing the names of the use case. Actors are represented using lines with the name of the actor written below the line. To represent an actor's participation in a system, a line is drawn between the actor and the use case.

Basware main artifact items are use cases, graphical user interface layouts, technical speficications, and features. Project team decided to emphasize the importance of use case. Use case template was rewritten by the project team members. UI related section was removed from the template and Alusta system analyst were trained with the new model. As some amount of detailed specifications were already written in a form of old feature, it was agreed that ID project team didn't require specifications to be rewritten and let feature element exist if that was used. The main message was still delivered to analysts, only use cases together with supporting screen elements and technical specifications was recommended combination which should be delivered to as assignment tasks for development teams.

5.5.2 Technical specification

Technical specification often captures high level architectural concepts but at the same can contain for example detailed database structural instructions. Service or subsystem descriptions and relations between these cover logical architectural issues. Technical specification may also operate as a reminder for general architectural styles and design patterns.

The main responsibility for technical specifications is on architects who should be able to answer for example performance, availability, security, and compliancy related questions. Technical specification supports team to get thing right. Interface descriptions and configuration examples are highly appreciated by team. Special attention must also be paid to user management-, authentication-, and authorization areas.

5.5.3 User Interface element

User Interface (UI) is the means by which the user and a solution, application, service or system, interact. User's interaction with the solution and its functions form the user experience, together with user's prior history and knowledge to operate a system.

User Interface design does not intuit anything on its own, it is the users who intuit the design. Therefore a vital part of user interface design is to know the target group's level of capability and expectations. User studies are conducted as part of UI design. When the user's current knowledge and the target knowledge are near equal, the usability of the solution is successful.

Basing the UI and the user experience design on worldwide usability standards, such as WCAG guidelines and ISO standards, secures that the accessibility and usability of the solution or service is successful to all user groups. The UI design must be based on understanding the business needs and requirements, defined in a functional specification. UI design is made simultaneously with specifications, user stories and use case definitions.

Additionally, understanding the technology of a system is part of UI design. Each system, service or solution is on a particular platform or a set of platforms. Design must be based on the building elements, UI controls and structures used in the target platform(s). UI and UX designers therefore work with product managers, system analysts, architects and developers during the design and implementation process of the user interface.

5.5.4 Understandable and high quality assignment list for development

To ensure clarity and buy-in, team should be aware of the sprint content in good time. One way to do this is to involve team to the grooming activities. Several reasons speak for the early sprint goal distribution. It creates alignment among product owner and the team. It minimizes variation by limiting the type of requirements worked on in a current sprint, for example, by choosing items from the same area. This improves

teamwork and can help increase velocity. Communication to stakeholders often comes easier when everybody knows what team is doing. (Pichler 2010, 59.)

When new developers to join the team, even if they are highly talented, they still need a lot of time to understand the system. In many cases, the existing developers need to reduce their programming time and spend it on training the new developers. Good documentation reduces the time spent here. Much of this heartache comes from a fundamental mistrust and fear of documentation. In a sense, some fear is justified because projects can get overwhelmed with documentation, but that is no justification to swing from one extreme to the other (Krisna 2007.) Objective should be to model just enough up front to ensure that development team has a mutual understanding of the solution approach and can start the work in a common and cohesive way (Collier 2012.)

During the ID project, understanding towards agile requirements management practices and methods grew steadily. Too detailed specifications up front were no further needed. Pilot projects product management started release preparation for the next release couple of months before planned development start date. This time the focus was on the business features and user stories. Pilot project's chief architect kicked of the user story estimation work. This quickly led to user story splitting work and further analysing the requirements stack.

Specifications should reflect what attributes are needed to enable people execute their work efficiently and effectively. Quality must be fitness for use. Specification mistakes and gaps should be minimized. But as specifications tend to change, written specifications should be checked for consistency, completeness, and correctness. Keeping specifications up-to-date, they should be as brief as they possibly can to encapsulate the facts and satisfy any standards of contractual requirements. (Evans 2004, 83-82.)

5.5.5 Dependencies and relationships

When managing requirements, focus is primarily on the business need or opportunity the requirements will address. Attention is usually paid to how requirements are formu-

lated, and whether they are clear, comprehensive and aligned to the project goals. There's nothing wrong with focusing on the "what" of requirements — that is, what is asked to be delivered. But to avoid any major problems during the project, it's also important to identify the related assumptions, constraints, dependencies and risks. (Haus 2012.)

Requirements management application, used in Basware, allowed already dependency creation between different requirement / specification items but it wasn't used in general. It is possible to create upstream, downstream, and co-dependent relationships between items. Upstream dependency item means that this particular item cannot be implemented without the following items being implemented first. Downstream dependency item tells that the following items cannot be implemented without this one being implemented first. Co-dependency item indicates that the following items have co-dependencies.

The Pilot project started to use dependencies between artifact items and user story items. User story item has artifact item(s) set as upstream dependencies. This enabled both product- and project management to easily see the current readiness status of the user stories. Product manager was only interested of knowing release status by looking at user stories and this approach served that need perfectly. It was also easier for system analysts to follow the implementation status of the user stories.

5.6 Results

After several iteration rounds and discussion in the the ID project team but also in the Pilot project team, a new requirements hierarchy was launced and introduced to target projects and other relevant stakeholders:

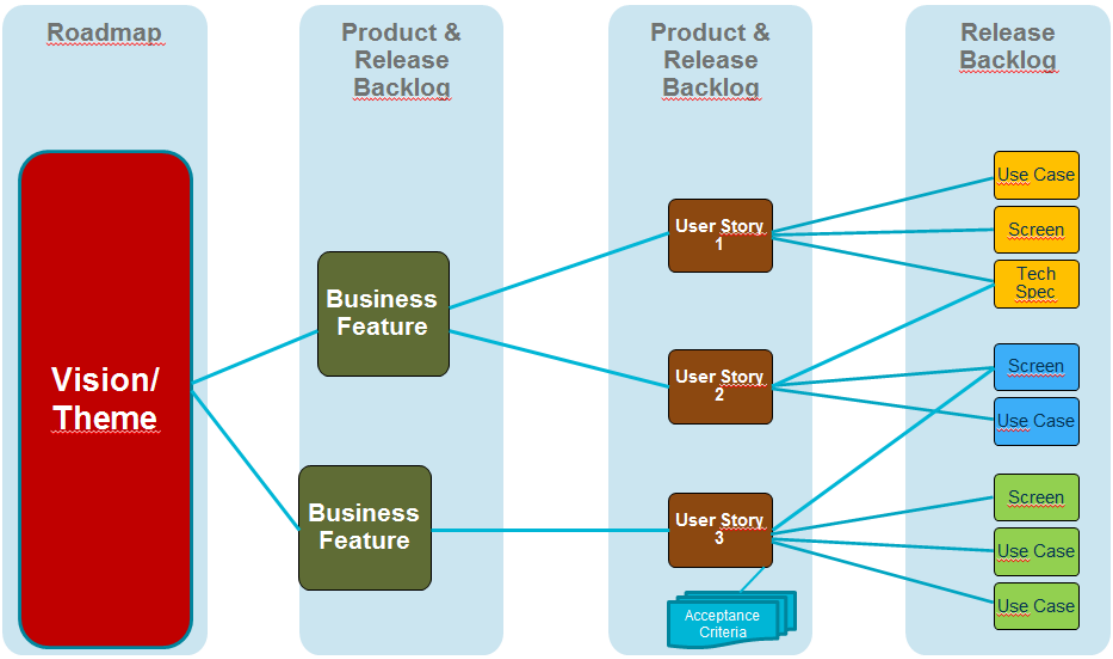


Figure 13. New requirements hierachy at Basware

According to Figure 13, product backlog contains all the requirements that are known, except detailed specifications. Release backlog on the other hand contains more detailed requirements that will be moved to implementation. Product management is responsible of constantly updating and prioritizing the backlog. This meant adding, removing, and specifying items in more detailed level.Visibility, transparency, predictability and control has increased a lot. Control over entirity especially on what comes to requirements management had significantly improved.

The new requirements hierarchy introduction was kept in several separate meetings having representatives from following stakeholders:

- System analysts
- Product managers
- Project managers
- Architects
- User experience experts
- User assistance and localization experts
- System testing team

System analysts' were introduced to the new structure in a separate system analyst meeting. Business features and user stories were explained by using training material created be the internal development team. As the old requirements hierarchy model constructed of a one level structure, artifacts, these were also discussed. Internal development team proposed that focus should be kept on use case-, screen-, technical specification-, and enhancement- elements from now on.

Chief architect was asked to present and communicate the new requirements model to program architects. An hour successful meeting was held and result was very positive. Meeting had an observer from the ID project team who also gathered notes from the meeting. Product managers also had their own introduction session which was held by senior product manager working for the Pilot project.

6 Requirements management community and roles

Basware's Alusta program product line management teams consist of product manager(s) and system analyst(s). A single project management team is comprised of a project manager, chief architect, application architects, product manager, and system analysts. During the ID project, a common Alusta project meeting practice was accepted and deployed across projects. Weekly project management team meetings were instituted and this new meeting practice brought together project members from different teams whose shared responsibility had been decision making inside teams.

It is essential to remember that no product, and no organization, can be infinitely agile in all dimensions. Finding the balance is the key to agility, but as there isn't any right formula, it isn't easy. Skill, talent, and knowledge breed quickness but compelling people to be faster breeds hurrying. Agility balances structure and flexibility, dependence and autonomy. (Highsmith 2004, 257-258).

Rather that scheduling with traditional charts, the project management team has the ability to create a plan that is populated with frequent milestones to keep the focus on

the overall objective. Nobody should worry about the change approval process but rather worry about change-tolerant design practices. (Poppendieck & Poppendieck 2006, 116.)

6.1 Reorganizing management – modern agile model

When a developer has a question, how much activity does it take to find out the answer? Are people at hand to help with a technical problem? Is the customer or customer representative readile accessible to answer a question about features? (Poppendieck M. & Poppendieck T 2003, 7). The primary target is to make sure that the business, especially the product owner, is able to clearly articulate what needs to be implemented and define what is of high quality. The requirements management cycle follows a scrum-like process that mirrors the development cycle while staying few steps ahead. The goal must be that requirements can be thoroughly vetted, organized, and communicated in a manner that is iterative, timely, and quality-focused. (Moccia 2012.)

Cottmeyer (2009) argues that agile product owner is a mix of many of our traditional roles in software development. The product owner is a combination of the product manager, the project manager, the analyst and a series of other business stakeholders. Exploration shows that this is an important job and materially different from the role of the product manager. It has been said that not many people have the breadth of skills for this role or the time available to perform the function well. Often a business analyst is inserted between the development team and other stakeholders in an attempt to provide the day to day direction.

The problem with this approach is that the analyst, or any other role aside from the product owner, is not held accountable appropriately because he/she does not "own" the product. A proxy can give the team direction but is not responsible for making the critical decisions when things need to be changed. The team is often required to seek separate permissions, often from the real product owner. Separating the real product owner from the team will slow the team down and increase the risk of developing in-appropriate or ineffective solutions. A shared accountability model is needed, which

could involve a team of people, each of whom work together as part of the team and are all accountable for delivering on the role of product owner. Cottmeyer (2009.)

Highsmith (2004, 58) brings in the project management point of view. Leadership-collaboration management drives managers to help teams balance at the edge of chaos. Finding the right balance for creating adequate documentation and for example requiring an adequate amount of up-front architecture work is the key of management. For development projects it is typical that teams face anxiety, change, ambiguity, and uncertainty along the way. Different styles of project management and various patterns of team operation are needed.

Leadership-collaboration management style enables organizations and teams to face the volatility of their environment. Experience makes it easier to identify right practices and to understand issues. The primary goal is to set right direction, provide guidance, and to facilitate connection among people and teams. Egalitarianism, competence, self-discipline, and self-organization are concepts that agile project management values resonates with. (Highsmith 2004, 58.)

Basware's new organization Products and Services together with R&D identified the need of rethinking roles. The ID project presented similar conclusions to management, based on theoretical study, and offered support to the role building phase if that was needed. Role specific work was about to take place in the end of H1 2013 but due to critical times in Alusta program, it never started.

6.2 Product Manager is not Product Owner

Common mistakes with the new product owner role entering an organization can be avoided by identifying problems, not least of which is ensuring the product owner receives an appropriate amount of management attention. Sponsorship should also come from the right level or from the right person. Management needs to trust the product owner and delegate decision making authority. The product owner should not be overworked because this easily leads into a situation where product owner becomes the bottleneck and limits the project's progress. (Pichler 2010, 18.)

Overworked product owners are apt to neglect grooming sessions, miss sprint planning and demo meetings and at times may not be available for questions only after a long delay. The team must support product owner because product owner work is carried out collaboratively. The product owner is a full-time job. (Pichler 2010, 18.)

Leffinwell (2012, 51) describes in short the product owner role elements:

- works with product managers, analysts, customers and other stakeholders to determine requirements
- maintains the backlog and sets priorities based on relative user value
- sets objectives for an iteration
- elaborates stories, participates in progress reviews, and accepts new stories

Pichler (2010, 4-6, 17-19) states that the product owner should be a visionary, a doer, a leader, a team player, a communicator, a negotiator, be empowered, committed, available, and qualified. At the same time the product owner must not be underpowered, overworked, partially committed, distant nor proxy owner.

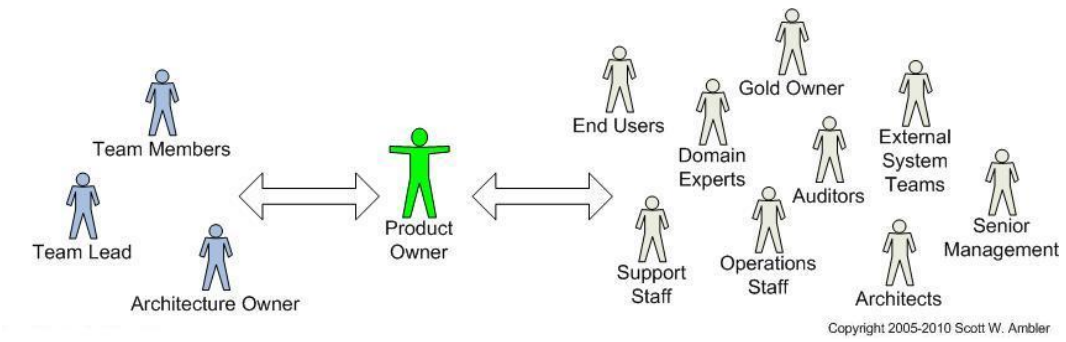


Figure 14. The role of product owner (Ambysoft Inc 2005-2012c)

As highlighted in Figure 14, as a stakeholder proxy, the product owner is the "go to" person for domain information. The product owner can provide timely information, make decisions and also help the team gain access to expert stakeholders. He/she educates the team in the business domain and is the gateway to funding. Product owner prioritizes requirements, defects, and other work items for the team. He/she facilitates

requirements modeling sessions, including requirements envisioning but is also an active participant in customer testing. (Ambysoft Inc. 2005-2012c.)

When representing the agile team to the stakeholder community, the product owner is the public face of team to project stakeholders. The product owner should be able to demo the solution to key stakeholders who weren't able to attend the normal iteration demo. He/she communicates team status, organizes milestone reviews, announces releases, and educates stakeholders in the development process. Priority, scope, funding, and schedule negotiations belong to product owner. (Ambysoft Inc. 2005-2012c.)

(Leffingwell 2012, 203) introduces some additional tactical activities for product owner:

- Setting objectives for the iteration
- Prioritizing and maintaining the backlog
- Participating in sprint planning meeting
- Elaborating stories on a just-in-time basis with the team
- Accepting stories into the baseline
- Accepting the sprint
- Driving release planning

The owner sets the vision and product objectives. Manages the ROI and works with marketing to position the product in the marketplace. The product owner is also a member of the team who works daily with the team to help the team to meet its objectives. It's quite obvious that in larger enterprise context, when Scrum is introduced, at times there is a mismatch between the method teachings and the existing organization's structure. (Leffingwell 2012, 203.)

A new practice was agreed inside the Pilot project team. System analysts should also participate in demo sessions arranged by other teams. The main purpose was to better share understanding of the current team states in the project. And as there were functional dependencies between development areas, new approach helped system analysts to continue discussing together about future challenges. After few weeks, a shared

demo session took place where teams presented their iteration results, one after another. The importance of product manager's presence in demo session was also understood better.

6.3 Community collaboration and communication

Customers want the system to deliver business value. Analysts or product managers help the customer articulate these valuable features in detail and communicate requirements to the development team. Developers should estimate the needed time and deliver working increments. Testers main target is to ensure that the system meets customer needs by creating comprehensive customer tests. This team has a shared purpose: Deliver business value. (Poppendiek & Poppendiek 2006, 107.)

Collaboration is an essential component of a healthy agile project community. Truly effective collaboration is probably the hardest thing to do well. Time-wasting meetings must be avoided but face-to-face communication must be the preference. Meetings should always have clear description with purpose. Colloborative sessions with ambiguous reasons should be avoided. The most rewarding sessions are those in which the issue gets addressed quickly with minimal time. Earlier than expected finishes are appreciated. Participants should be able to add value to the session but at the same time "the membership limitation" should be successful. Participants must focus on the purpose of the session. (Collier 2011, 69-71.)

According to Ambysoft Inc (2001-2012a) research called "communication strategies and effectiveness" it is justified to say that face to face communication is highly recommended between team members but also between team and stakeholders:

Communication Strategy	**Within Team**	**With Stakeholders**
Face to face	4,25	4,06
Face to face at Whiteboard	4,24	3,46
Overview diagrams	2,54	1,89
Online chat	2,1	0,15
Overview documentation	1,84	1,86
Teleconference calls	1,42	1,51

Videoconferencing	1,34	1,62
Email	1,08	1,32
Detailed Documentation	-0,34	0,16

Table 1. Effectiveness of communication strategies on agile development teams Ambysoft Inc (2001-2012a)

Answers, in Table 1, were rated on a scale of 1 to -5 (very ineffective) to +5 (very effective). An interesting message is that overview documentation was seen as being reasonably effective but detailed documentation was not. Online chat was thought to be effective between developers but not with stakeholders. This is probably a reflection of cultural differences and experiences between the two communities". (Ambysoft Inc, 2001-2012a).

6.4 Grooming the backlog

The intent of a "grooming" session is to ensure that the backlog remains populated with items that are relevant, detailed and estimated to a degree appropriate with their priority. This also helps everyone to share an understanding of the current project or product and its objectives.(Agile Alliance 2011).

"Prioritization is found in all the agile approaches. A common practice is to implement the highest priority features first to be able to deliver the most business value. During development, the understanding of the project increases and new requirements are added. To keep priorities up-to-date, prioritization is repeated frequently during the whole development process" (Paetsch, F & Eberlein, A & Maurer, F).

> Prioritizing requires deciding how important an item is. If everything is high-priority, everything is equally important. This means in effect that nothing is a priority, so there is only a slim chance of delivering what the customer really needs. Like the other grooming activities, prioritizing is best carried out be the entire Scrum team. This leverages the team's collective knowledge and generates buy-in. (Pichler, 2010.)

The Pilot project identified the need of grooming activity during the end of H1 2013. As grooming wasn't a common practice in the organization up to to that point it was

overtaken by other activities which were seen as priorities. Even product management didn't arrange common sessions where backlog items would have been viewed, discussed, and planned further together.

Prioritization directs development work by focusing individuals on the most valuable and important items. This brings flexibility into the process and allows delaying decision making about the lower-priority items. It also enables development and management teams to evaluate options longer, gather feedback and acquire more knowledge. (Pichler, 2010 quoting Poppendick 2003).

To get the best out of a backlog grooming meeting, Cohn (2013) reminds that, it probably comes down to the same few things for any meeting:

- Keep it as short as possible.
- Show up prepared.
- Encourage everyone to participate.

The Figure 15 overviews the approach to managing requirements where development team has a stack of prioritized and estimated requirements which need to be addressed. New requirements are prioritized by project stakeholders and added to the stack in the appropriate place. The backlog is initially filled as the result of requirements envisioning efforts at the beginning of the project. Project stakeholders have the right to define new requirements, change their minds about existing requirements, and reprioritize requirements as they see fit. Stakeholders are responsible for making decisions and providing information. On some projects an analyst is often in the role of a product owner. (Ambysoft Inc. 2005-2012a.)

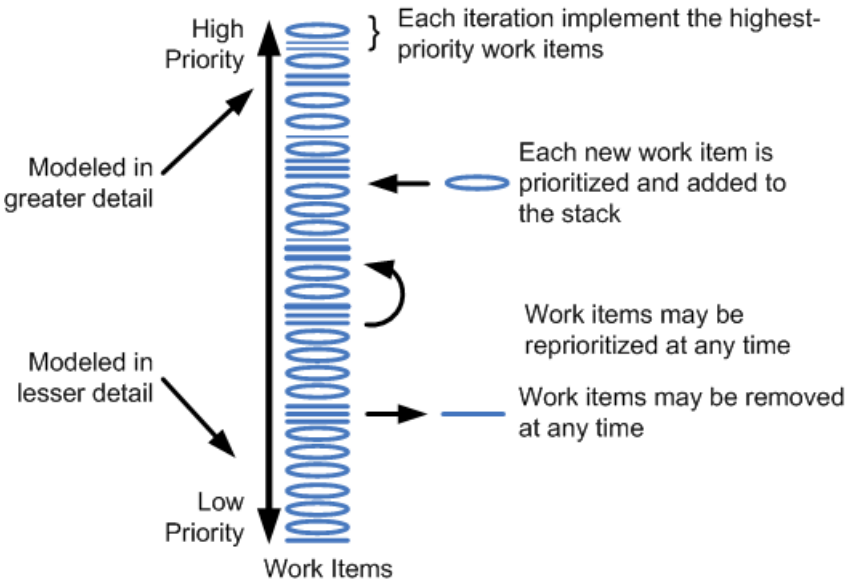


Figure 15. Scrum requirements management (Ambysoft Inc. 2005-2012a)

According to Pohl & Rupp (2011, 97-99), the goal of the review is to receive co-worker's expert opinion with regard to the quality of a requirement. By validating together, wrong assumptions and interpretations can be avoided more easily. While walking through requirements, quality flaws can be identified while everyone is gaining a shared understanding of the requirement. Everyone has also an opportunity to provide additional information to the group along with the actual requirement.

Questionnaire results, visible in Attachment 4, showed that backlog prioritization happens regurlarly in most of the target projects. On the other hand, clear review practices does not exist and only half of the respondents stated that review practice is in use in any form. The ID project team noted that review practices were not part of the work but actions were decided to move further to the future.

It is common that projects experience gaps in time between when requirements are defined and when development begins. It is important to note that this occurs on a regular basis. The larger the gap in time between definition of requirements and development, the greater the risk that the right product does not get developed. (Moccia, 2012.)

The risk of losing vital team members, knowledge, and overall team availability also gets higher in case of delay between specification work and development. Decomposition is a way to pick up where things left off, by using the product backlog to communicate and share requirements. Agility fosters continuous improvement, time-boxed development cycles, and more quickly delivering value to the end users. That value will be driven to a large extent by the quality and clarity of requirements that feed the software development process. A timely approach to requirements as the starting point will help to ensure that the process is optimized. (Moccia, 2012.)

7 Conclusions and discussion

No matter what the development methodology is, as Cockburn (2007, 140, 179) proposes, make refinements, maintain flexibility and create new process that may lead to a higher probability of success. Those learnings that are replicable and could be relevant for future development projects should be extracted and added to repertoire of tactics and strategies. By listening to other project teams when they describe their experiences and lessons learned is valuable. One of the most essential conventions is to identify and understand a team's methodology, as it is highly unlikely any methodology will work "straight out of the box". Too many complicated rules should be avoided and team should create a small handful of well considered guidelines that should be enforced rather than a series of overly complex, and potentially rules. In combination, this creates a framework for groups and their members to maintain discipline throughout a project but also maintains being nimble and flexible in the face of inevitable disruptions, e.g. technological change.

7.1 Conclusion of internal development project results

The new method, requirements hierarchy, was the most significant individual achievement by some margin. Results demonstrated that the requirements hierarchy enables and promotes systematic handling of requirements and the Pilot project team supported this in practice. Initially, the requirements hierarchy concept was missing and effectively a flat series of different types of requirements equal to one another served development teams. A new requirements hierarchy was easily understood but in practice the

mobilization was more challenging for the ID project team than originally anticipated in practice. ID project team didn't get enough management support for the change because of the overburden situation in the Alusta program. Management commitment was vague.

The new hierarchical structure was broadly approved by target projects but in practice, the Pilot project team was the only group implementing the actual transformation so far. The most common message from other Alusta projects was that the new method seemed to be very interesting and reasonable but no one had time to incorporate it into their existing processes. A limitation of its implementation during this thesis assignment was that the responsibility lay solely with the ID project team and deployment across all project teams was not possible in the timeframe provided. In hindsight, ID project's objective to implement the new method to every Alusta projects may have been too ambitious.

However, the ID project team performed well during the H2 of year 2012. Short term improvements were identified quickly and implemented into practice as planned. At the same time the questionnaire was created and interviewer was selected from the project team. Target roles and interviewees were defined in separate meetings. Schedule for the interview execution was also agreed. Timeline for the questionnaire was challenging but the interviewer managed to arrange one-on-one meetings with each of the selected interviewees. Valuable results were gathered and analysed.

Shortly after the project team started H1 2013 work, a significant organizational change took place at Basware. The project team was required to apply for a mandate to be able to continue its work because the project owner together with system analysts moved under a new organization and rest of the project team members stayed in the R&D organization. It was a shake-up for the ID project and its objectives because now requirements management got automatically wider attention between these two equal organizations and it was no longer only seen as a sub process for Basware's PLC model. Mandate was granted and project team continued its work after a short break.

The ID project owner was very busy with other duties in the company during the project and project management responsibilities were handed over to another team member in the middle of H1 2013. The ID project became more pragmatic and agile approach took an important role and kept the focus on the essential issues. Spirit inside the ID team was good and disagreements were managed in a professional way.

7.2 Personal evaluation of the value of research project

Academic study and author's role in the Pilot project enabled new ideas, theories and best practices to be taken into practice in a real world setting - something that is often lacking in the workplace. Author's personal interests together with genuine desire to develop requirements management practices contributed to finding better solutions and finally taking these into use. Unfortunately this didn't happen in all the target projects. On a positive note, the requirements management position and importance in the organization took on an elevated level of importance throughout the organization over the period of the study and will hopefully continue into the future as it continues to demonstrate value.

During the thesis work it was also highlighted that requirements management isn't only specification work but a substantial process that has the potential to have a strong influence on a product throughout its entire life cycle. As new findings and suggestions arose those were discussed and analysed inside the ID project team. Especially the idea of bringing in the new user story element was warmly welcome and belief in success grew.

Although ID project was not successful in implementing successful changes to all Alusta projects as planned, a positive outcome was that change towards a more systematic way of handling requirements evolved in the process. The partial implementation failure occurred in part due to insufficient planning ahead of the change. Certain changes that had been previously agreed were also not put into practice. The setup for changes within the Pilot project was more effective due to key team member's personal attitudes and support for the initiative and general openness to change.

Communication of theory was shared in an iterative way both among the ID project and the Pilot project. Findings and real world examples were discussed and considered and some finally adapted to practices. The theoretical framework helped participants understand the fundamental meaning and thought process behind the user stories and their position and value in requirements hierarchy.

Academic research also increased the author's motivation and led to more enthusiastic agile definition work. The theory assisted in developing deeper insights into topics and helped the author to step outside the present state and view things from wider perspective. The author also started to act more with courage and take a more prominent role in different matters. Clearly some sort of satisfaction for the existed requirements management state crumbled away. Tool-centered opinions turned into process-centered. In general the way of thinking requirements management practices and methods has changed which makes it easier to challenge issues but also make reasonable and valuable decisions.

7.3 Future research and continual improvement

Findings and requirements management improvements during this project created a solid baseline to diversify the usage of the new requirements hierarchy at Basware. New methods came into existence and were shown to work through the Pilot project. ID project team decided that the change work needed to be continued so that other target projects could start using the hierarchical approach in their everyday work with the Pilot project acting as the "prototype" for successful delivery. Ideally, resistance to change will be less prevalent than in the first iteration due to the encouraging results of the Pilot project, however continuing support from management will be essential going forward.

This thesis work did not address requirement estimation and possible techniques that could be utilized. It's an own piece of art and in the end still one the success factors for projects to be able to reach targets in time. By doing estimations poorly and careless, this often leads to eventual unexpected surprises.

One of the future improvement areas is to finalize requirement management process update work, a key element of which is changing employees thinking and approach to the subject. To ensure that other Alusta projects will start working by utilizing the new requirements hierarchy ID project must create a deployment plan for the change.

ID project team put its full resources behind the functional requirements when creating the new hierarchy, a result of which was non-functional requirements being temporarily handled with lower priority. It is also extremely important to turn focus more on the non-functional requirements side and include these stronger to requirements and specifications. As presented in this work, the acceptance criteria of a user story is one of the elements for this.

Hierarchical levels of requirements and their systematic usage in backlogs helps everyone to better understand and identify different parts of the whole product content. This also lays the foundation for product and project management towards effective change management. Requirements change management is a completely own area and needs attention in the near future. Change management is difficult and improvements in working practices are always needed. The new requirements hierarchy supports this objective and makes it possible to continue working effectively in spite of this challenge.

Currently, requirements management process and work is not directly measured. The importance of measurement was discussed inside the internal project team several times, however a clear set of metrics agreed by everyone has not yet been developed. This topic was postponed to the near future.

This thesis work brought value to requirements management process but most importantly it tied project management members and other stakeholders closer to each other. Effective co-operation between stakeholders is one of the key elements towards success. Grooming meetings should be taken into use everywhere because grooming session enables people to validate and review requirements together. Development teams should be included to grooming sessions as soon as possible. One additional

finding was that shared sessions with individuals who's responsibility is to turn the specs into a working application are needed.

7.4 Discussion

Software development agility is constrained by a lack of requirements management agility. Constrained knowledge of the agile requirements management slows down the work and is often a primary reason for bad quality requirements. Results of this study provide evidence and knowledge to requirements management practices and opens up the possibility to further utilize this inside the entire organization. Findings during the project expanded the conception of the hidden requirements management challenges. The importance of open and transparent communication was highlighted and discussed several times. The evidence from the interviews supported the current understanding of the challenges that product management and other stakeholders had.

The importance of proper requirements management in the beginning of system development should be better communicated across the organization. Requirements management is about commitment and a disciplined, systematic -way of working. By understanding benefits of well managed requirements, it helps to create a systematic working environment and effective results. A longer term objective is for each and every individual involded in development (in advance, during or after) to understand the importance of requirements management and how it can make them more effective in their day to day job. Key elements include documentation, clear task delivery, efficient communication, following common guidelines and above all: co-operative mission.

It is not possible to specify requirements which will never change during the life time of the system or even over the course of the development project. Agile development enables and allows relatively quick changes to take place during different development project phases. Requirement change management must be taken into account when development work faces challenges or a situation simply demands it.The role of product management is not only relevant in managing customer needs but also in managing other stakeholder needs such as development team. High quality requirements and

specifications reduce changes but don't guarantee outcome quality. Effective communication is very important between management and the development team. Everyone must share an understanding of what needs to be built - shared responsibility at every stage.

Continual requirements scoping is an essential part of project success. If scoping is unsuccessful, it will also be a challenge to maintain the schedule. Scoping is not an easy task and experience helps to get the right effort estimates. Constant negotiation is needed in deciding what to include and what is left out. Understanding the importance of smooth co-operation is the key to success - together we cannot fail.

Bibliography

Accept Software. 2012. URL: http://www.accept360.com/company. Accessed: 10 Aug 2013.

Agile Alliance, 2011. Backlog grooming. URL: http://guide.agilealliance.org/guide/grooming.html. Accessed: 19 Aug 2013.

Alexander, I. F. & Stevens, R. 2002. Writing Better Requirements. Pearson Education. Harlow.

Ambler, S.W. 2009. Requirements specifications on agile projects. URL: http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/923/Requirements-Specifications-on-Agile-Projects.aspx. Accessed: 26 May 2013.

Ambysoft Inc. 2001-2012a. Communication on Agile Software Projects. URL: http://www.agilemodeling.com/essays/communication.htm. Accessed: 1 Sept 2013.

Ambysoft Inc. 2001-2012b. Agile Requirements Modeling. Ambysoft Inc. URL: http://www.agilemodeling.com/essays/agileRequirements.htm. Accessed: 13 Mar 2013.

Ambysoft Inc. 2005-2012a. Agile Best Practice: Prioritized Requirements. Ambysoft Inc. URL: http://www.agilemodeling.com/essays/prioritizedRequirements.htm. Accessed: 4 Apr 2013.

Ambysoft Inc. 2005-2012b. Agile Requirements Change Management. Ambysoft Inc. URL: http://www.agilemodeling.com/essays/changeManagement.htm. Accessed: 14 Nov 2012.

Ambysoft Inc. 2005-2012c. The Product Owner Role: A Stakeholder Proxy for Agile Teams. Ambysoft Inc. URL: http://www.agilemodeling.com/essays/productOwner.htm. Accessed: 23 Apr 2013.

Beck, K. & Fowler, M. 2001. Planning extreme programming. Pearson Eduction, Inc. Boston.

Boehm, B. & Turner R. 2004. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. URL: http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/Agile/21630718.pdf. Accessed: 6 June 2013.

Boost New Media. 2010. User Stories: a beginners guide to acceptance criteria. URL: http://www.boost.co.nz/blog/agile/acceptance-criteria/. Accessed: 15 May 2013.

Bryman, A. & Bell, E. 2011. Business research methods. 3rd edition. Oxford University Press Inc. New York.

Cockburn, A. 2007. Agile software development: the cooperative game. 2nd edition. Pearson education, Inc. Boston.

Cohn, M. 2008. Succeeding with Agile - Mike Cohn's Blog: Advantages of the "As a user, I want" user story template. URL: http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template. Accessed: 10 Jun 2013.

Cohn, M. 2013. Backlog Grooming: Who Should Attend and How to Maximize Value. URL: http://www.mountaingoatsoftware.com/blog/backlog-grooming-who-should-attend-and-how-to-maximize-value. Accessed: 11 Jul 2013.

Cohn, M. 2012a. User stories for Agile development. Video. URL: http://vimeo.com/43601248 . Accessed: 4 Jul 2013.

Cohn, M. 2012b. Agile Succeeds Three Times More Often Than Waterfall. URL: http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall. Accessed: 14 Jul 2013.

Collier, K. 2012. Agile Analytics: A value-driven approach to business intelligence and data warehousing. Pearson Eduction, Inc. Boston.

Costello, P.J.M. 2003. Action Research. Continuum. London.

.

Costley, C & Elliot, G & Gibbs, P. 2010. Doing work based research: approaches to enquiry for insider-researchers. SAGE Publications Ltd. London.

Cottmeyer, M. 2009. The Product Owner Team. URL: http://www.leadingagile.com/2009/03/the-product-owner-team/. Accessed: 19 Aug 2013.

Evans, I. 2004. Achieving software quality through teamwork. Artech House, Inc. Norwood.

Garret J. J. 2002. The elements of user experience: User-centered design for the web. A Division of Pearson Education. New York.

Haus M. 2012. Gather More Than Just Requirements. Blog. Project Management Institute. URL: http://blogs.pmi.org/blog/voices_on_project_management/2012/12/gather-more-than-just-requirem.html. Accessed: 30 July 2013.

Highsmith J. 2004. Agile project management., creating innovative products. Pearson Education, Inc. Boston.

Jackson, W. 2013. What Characteristics Make Good Agile Acceptance Criteria? Segue Technologies. URL: http://www.seguetech.com/blog/2013/03/25/characteristics-good-agile-acceptance-criteria. Accessed: 5 Sept 2013.

Lahanas, S. 2013. Agile or Not, Requirements Management Can't Be Skipped. URL: http://news.dice.com/2013/01/03/why-requirements-still-matter/. Accessed: 9 Mar 2013.

Leffingwell, D. 2011. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series). Pearson Education, LTD. Boston.

Lawrence, R. 2009. Patterns for Splitting User Stories. URL: http://www.richardlawrence.info/2009/10/28/patterns-for-splitting-user-stories/. Accessed: 20 Aug 2013.

Krisna. 2007. Agile development and requirements management. URL: http://www.thoughtclusters.com/2007/09/agile-development-and-requirements-management/. Accessed: 30 July 2013.

Maciaszek, L.A. 2005. Requirements analysis and system design. Pearson Education Limited. Harlow.

Martin, R.C. 2012. Agile Software development: principles, patterns, and practices. Pearson Education, INC. New Jersey.

Moccia, J. 2012. Agile Requirements Definition and Management. Scrum alliance. URL: http://www.scrumalliance.org/community/articles/2012/february/agile-requirements-definition-and-management. Accessed: 2 Jul 2013.

Paetsch, F & Eberlein, A & Maurer, F. 2003. Requirements Engineering and

Agile Software Development. Article. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.15.3395&rep=rep1&type=pdf. Accessed: 22 Aug 2013.

Pichler, R. 2010. Agile product management with scrum, creating products that customers love. Pearson Education, INC. Boston.

Pohl, K. & Rupp, C. 2011. Requirements engineering fundamentals. Rocky Nook Inc. California.

Poppendieck, M. & Poppendieck, T. 2006. Lean software development, and agile toolkit. Pearson Education Corporate Sales Division. New Jersey.

Pund, P. 2013. Should Defects Be Considered User Stories? Scrum alliance. URL: http://www.scrumalliance.org/community/articles/2013/2013-may/should-defects-be-considered-user-stories. Accessed: 10 Aug 2013.

Rally Software. 2013. Write a Great User Story.URL: https://prod.help.rallydev.com/writing-great-user-story. Accessed: 5 Aug 2013.

Robertson, S. & Robertson J. 2012. Mastering the requirements process: Getting requirements right. Pearson Education, LTD. Boston.

Robertson, S. & Robertson J. 2006. Mastering the requirements process. Second edition. Pearson Education, Inc. Boston.

Software Engineering Institute 2010. Technical Report. CMMI® for Development, Version 1.3. Improving processes for developing better products and services. URL: http://cmmiinstitute.com/assets/reports/10tr033.pdf. Accessed: 10 Dec 2012.

Scrum.Org. 2009. What is Scrum? URL: https://www.scrum.org/Resources/What-is-Scrum. Accessed: 24 Sept 2013.

Scrum Alliance. 2013. Why Scrum? URL: http://www.scrumalliance.org/why-scrum. Accessed: 3 Oct 2013.

Techopedia. 2010-2013. Use case. URL: http://www.techopedia.com/definition/25813/use-case. Accessed 15 Sept 2013.

VersionOne. 2013. Benefits of Agile Software Development. URL: http://www.versionone.com/Agile101/Agile-Software-Development-Benefits/. Accessed: 15 Oct 2013.

Wikipedia. 2006. Action Research. URL: http://en.wikipedia.org/wiki/Action_research. Accessed: 30 Oct 2013.

Attachments

Attachment 1. Closed Questionnaire

Long term planning

- Clear portfolio planning inside unit
- Product line vision clear
- Future release themes defined
- Basware strategies visibly taken into account in release planning
- Backlog visible and prioritized X months to the future
- Backlog items estimated X months to the future

Project beginning planning

- Supported user stories (end-to-end high level use cases) defined/refreshed in beginning of project
- Supported business features defined/refreshed in beginning of project
- Supported process flows defined/refreshed in beginning of project
- Navigation diagrams defined/refreshed in beginning of project
- Configurable elements in system defined/refreshed in beginning of project
- Data model defined/refreshed
- Business document statuses defined/refreshed (PO, Invoice etc…)
- Req Spec reviews executed with System Analyst
- Req Spec reviews executed with Architect
- Req Spec reviews executed with Product Manager
- Req Spec reviews executed with Test Manager
- Req Spec reviews executed with UX
- Req Spec reviews executed with UA
- Req Spec reviews executed with Consulting
- Req Spec reviews executed with Support
- Req Spec approved before implementation start
- Non-func requirements defined/refreshed
- Feature specification content clear and unambiguous
- Feature specification testability verified

Continuous product management

- Customer input traceable to product version
- Release baselining P1
- Release scope ranked and estimated P1
- Release baselining P2
- Release scope ranked and estimated P2
- Release baselining Final outcome
- Release Requirements document
- Comprehensive release scope
- Sprint contents planned before sprint planning
- Sprint contents planned at least 1 sprint in advance
- Sprint contents planned at least 2 sprint in advance

Specification practices

- Use Cases / extension (alternative flows) used
- UI layouts and table of component contents utilized
- Ranking-Release constantly executed
- Requirement Dependencies exploited
- Defect mapping to Requirements done
- Baseline defined for requirement change
- Baseline defined for requirement reviews
- UA approved terminology used in specifications

Project Management perspective

- Specification work estimated in project
- Specification progress tracked
- Specification work included in sprints
- Technical design estimated in project
- Technical design progress tracked
- Technical design included in sprints

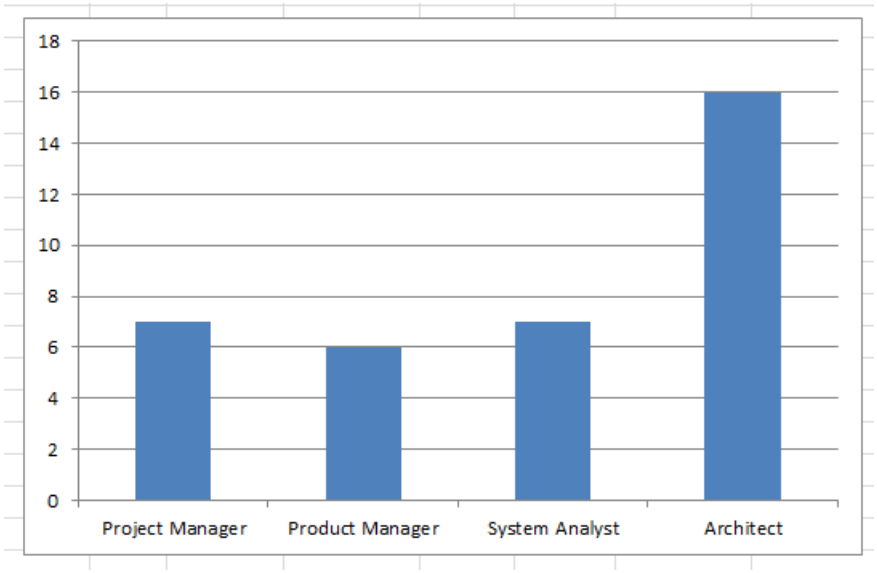
Requirements reuse scenarios

- Requirements traced to test cases
- Release notes created from requirements
- System descriptions created from requirements

Answer options for statements and questions:

- No
- Sounds right but No
- Partially Yes
- Yes
- We have other approach

Attachment 2. Interviewees by role

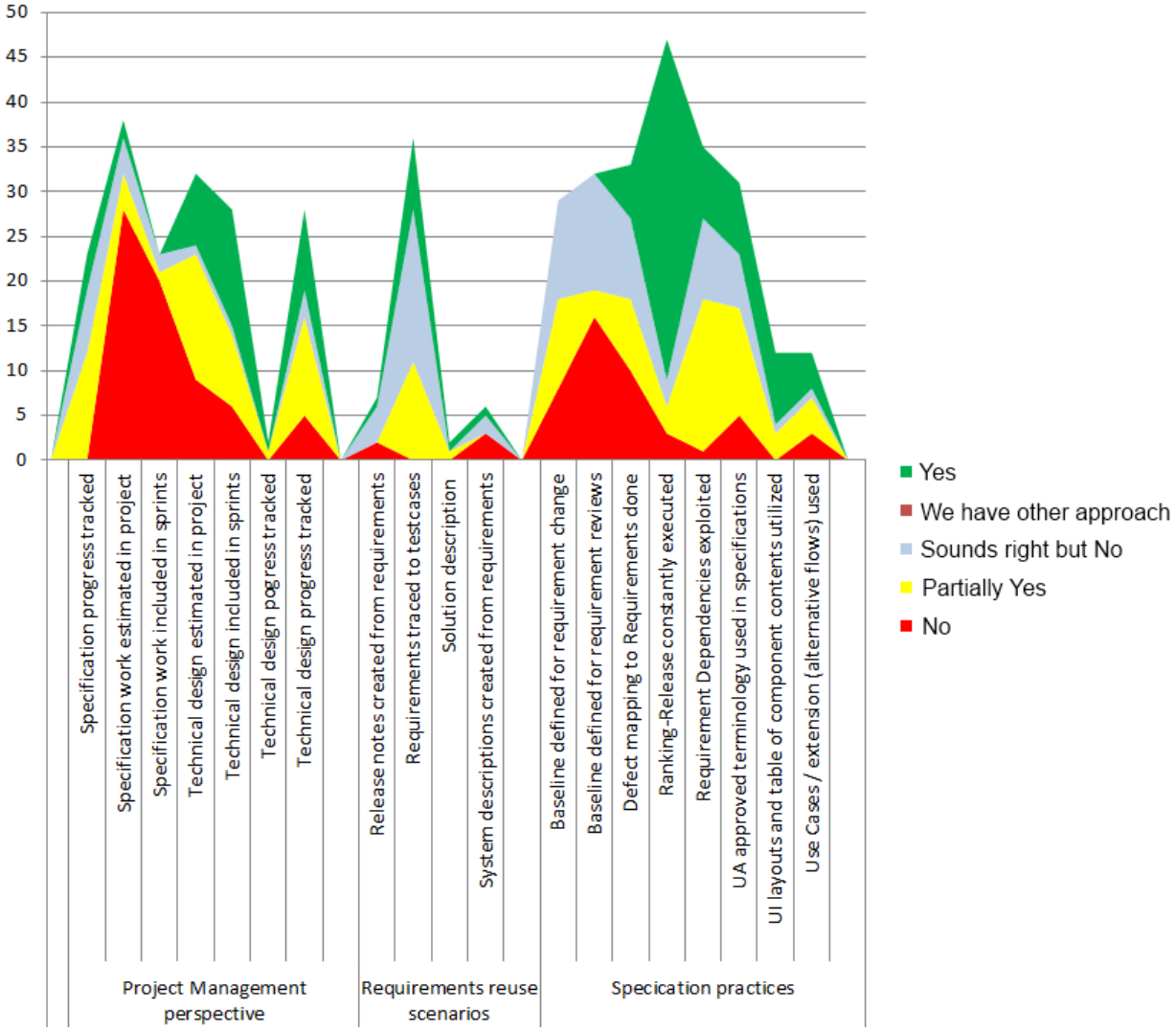


Attachment 3. Questionnaire results in figures

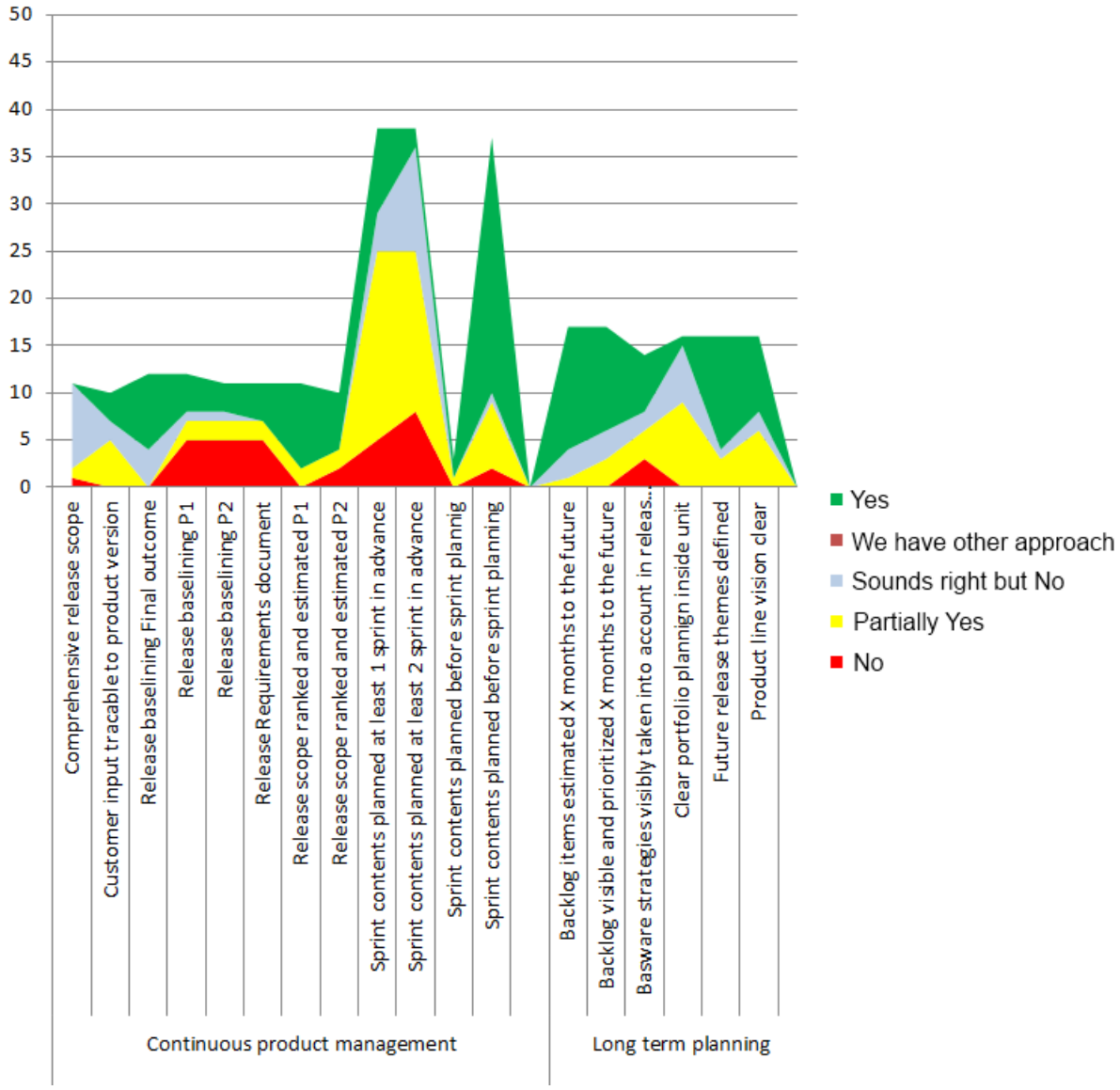
Questions / statements	No	Sounds right but No	Partially	Yes	Grand Total
Continuous product management	**33**	**33**	**61**	**77**	**204**
Comprehensive release scope	1	9	1		11
Customer input tracable to product version		2	5	3	10
Release baselining Final outcome		4		8	12
Release baselining P1	5	1	2	4	12
Release baselining P2	5	1	2	3	11
Release Requirements document	5		2	4	11
Release scope ranked and estimated P1			2	9	11
Release scope ranked and estimated P2	2		2	6	10
Sprint contents planned at least 1 sprint in advance	5	4	20	9	38
Sprint contents planned at least 2 sprint in advance	8	11	17	2	38
Sprint contents planned before sprint planning			1	2	3
Sprint contents planned before sprint planning	2	1	7	27	37
Long term planning	**3**	**17**	**25**	**51**	**96**
Backlog items estimated X months to the future		3	1	13	17
Backlog visible and prioritized X months to the future		3	3	11	17
Basware strategies visibly taken into account in release planning	3	2	3	6	14
Clear portfolio planning inside unit		6	9	1	16
Future release themes defined		1	3	12	16
Product line vision clear		2	6	8	16
Project beginning planning	**45**	**94**	**125**	**161**	**425**
Business document statuses defined/refreshed (PO, Invoice etc…)	6	6	3	21	36
Configurable elements in system defined/refreshed in beginning of project	4	7	12	8	31
Data model defined/refreshed		2	12	15	29
feature specification content clear and unambiguous	4	4	16	21	45
feature specification testability verified	2	22	10	11	45
navigation diagrams defined/refreshed in beginning of project	3	4	7	7	21
Non-func requirements defined/refreshed	1	5	15	10	31
Req Spec approved before implementation start	3	6	6	7	22
Req Spec reviews executed with Architect		2	7	6	15
Req Spec reviews executed with Consulting	4	1	4	1	10
Req Spec reviews executed with Product Manager		6	2	8	16
Req Spec reviews executed with Support	5		4		9
Req Spec reviews executed with System Analyst	1	5		6	12
Req Spec reviews executed with Test Manager	1	9	1	2	13
Req Spec reviews executed with UA	4	1	4	4	13
Req Spec reviews executed with UX	1	5	2	8	16
supported business features defined/refreshed in beginning of project	1		2	11	14
supported process flows defined/refreshed in beginning of project	3	9	12	8	32

supported user stories (end-to-end high level use cases) defined/refreshed in beginning of project	2		6	7	15
Project Management perspective	**68**	**18**	**51**	**37**	**174**
Specification progress tracked		7	12	4	23
Specification work estimated in project	28	4	4	2	38
Specification work included in sprints	20	2	1		23
Technical design estimated in project	9	1	14	8	32
Technical design included in sprints	6	1	8	13	28
Technical design pogress tracked			1	1	2
Technical design progress tracked	5	3	11	9	28
Requirements reuse scenarios	**5**	**23**	**12**	**11**	**51**
Release notes created from requirements	2	4		1	7
Requirements traced to testcases		17	11	8	36
Solution description			1	1	2
System descriptions created from requirements	3	2		1	6
Specication practices	**46**	**53**	**60**	**72**	**231**
Baseline defined for requirement change	8	11	10		29
Baseline defined for requirement reviews	16	13	3		32
Defect mapping to Requirements done	10	9	8	6	33
Ranking-Release constantly executed	3	3	3	38	47
Requirement Dependencies exploited	1	9	17	8	35
UA approved terminology used in specifications	5	6	12	8	31
UI layouts and table of component contents utilized		1	3	8	12
Use Cases / extension (alternative flows) used	3	1	4	4	12
Grand Total	**200**	**238**	**334**	**409**	**1181**

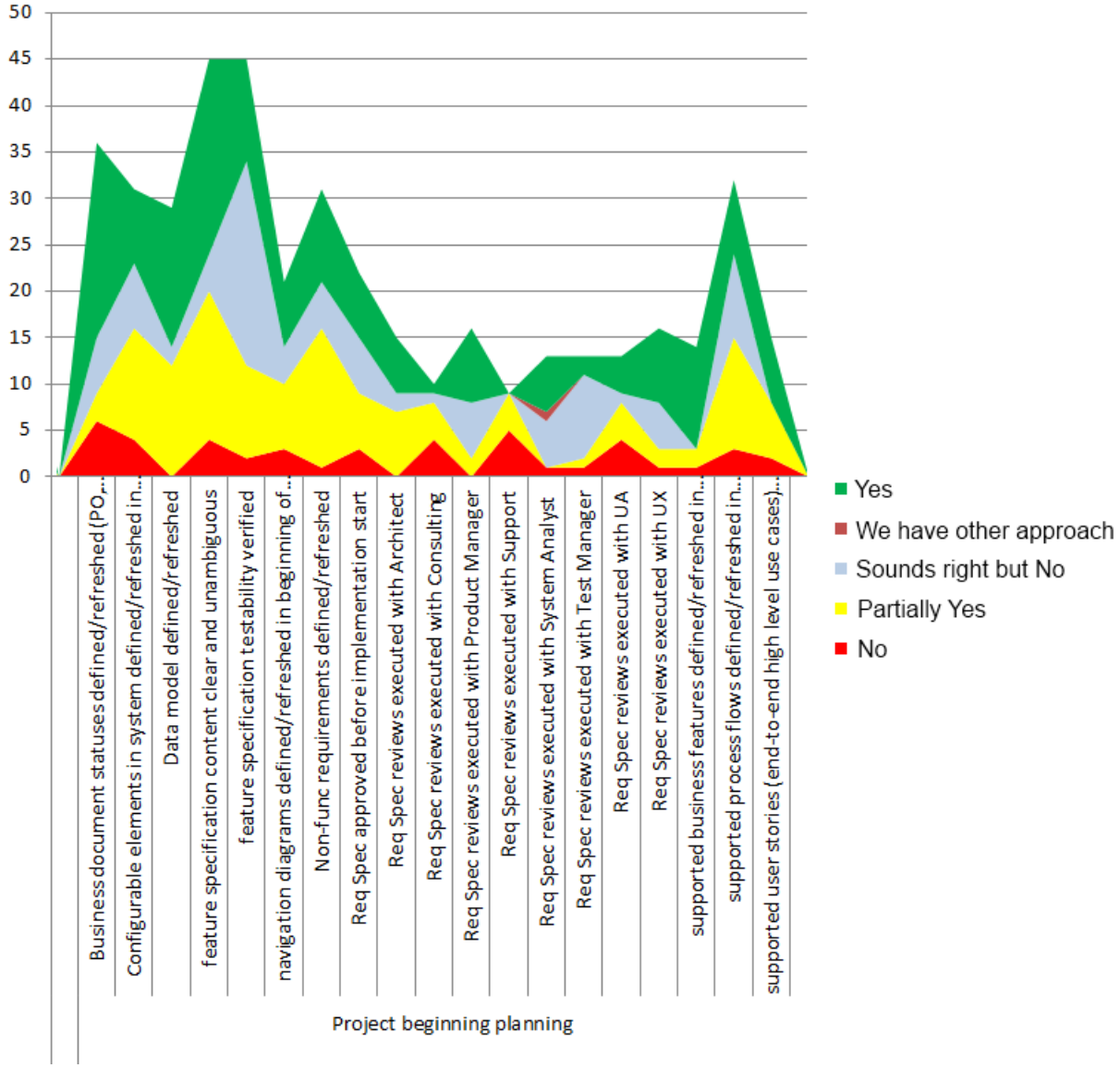
Attachment 4. Visual presentation of the questionnaire results, part 1



Attachment 5. Visual presentation of the questionnaire results, part 2



Attachment 6. Visual presentation of the questionnaire results, part 3



Attachment 7. Basware's requirements management workflow diagram