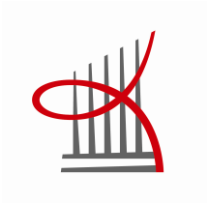

Improving Quality in Evolving Software Development Team Practices

Annikki Jussila

Master's thesis
September 2013
Information Systems
Competence

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojärjestelmäosaamisen koulutusohjelma, ylempi AMK

JUSSILA ANNIKKI:
Kehittyvän ohjelmistokehitystiimin laadun ja käytäntöjen parantaminen

Opinnäytetyö 63 sivua
Syyskuu 2013

Opinnäytetyön taustalla on pari vuotta sitten perustetun tiimin tavoite parantaa laatua toimintatapoja kehittämällä. Laadun parantaminen tähtää lopputuotteen ominaisuuksiin ja virheettömyyteen sekä asiakkaan tarpeiden täyttämiseen. Tähän voidaan välillisesti vaikuttaa myös niillä prosesseilla ja käytännöillä, joiden avulla tuotetaan haluttu lopputulos tehokkaasti. Opinnäytetyön tavoitteena olikin löytää ja kehittää laadun parantamiseen tähtääviä käytäntöjä kyseisessä tiimissä.

Organisaation toiminta perustuu ohjelmiston kehitysprojekteihin. Projektien hallinta organisaation tasolla toteutuu perinteisen projektinhallinnan keinoin, ja tiimissä käytetään ketteriä menetelmiä (agile ja scrum). Ohjelmistoa kehitetään aikaisemman version perusteella, joten projektit toistuvat samankaltaisina. Tästä syystä prosessien kehittäminen ja hallinta ovat myös viitekehyksenä käytäntöjen kehittämiselle.

Käytäntöjen kehittäminen on koko tiimin yhteistyötä, joten työn toteutus perustui toimintatutkimukseen ja havainnointiin. Löydetyt menetelmät kuvattiin tavalla, joka kertoo, miten ne on juuri tässä tapauksessa toteutettu ja koettu hyödyllisiksi. Tutkimuksen ei ole ollut tarkoitus olla yleispätevä, mutta löydettyjä käytäntöjä voidaan soveltaa muissakin ympäristöissä. Lisäksi kokonaisuuden avulla saatiin havainnollistettua muutosten toteuttamista.

Tutkimuksen tuloksena saatiin uusia käytäntöjä eri tarkoituksiin ja projektin eri vaiheisiin. Laadun parantaminen on kuitenkin jatkuvaa työtä: löydettyjä käytäntöjä pitää muokata edelleen ja uusia toimintatapoja kehittää sekä tiimin tasolla että yhteistyössä organisaatiotasolla.

Asiasanat: laadunhallinta, prosessijohtaminen, ohjelmistokehitys, ketterät menetelmät

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Master's Degree in Information System Competence

JUSSILA ANNIKKI:
Improving Quality in Evolving Software Development Team Practices

Master's thesis 63 pages
September 2013

This thesis deals with a software development team's desire to improve quality by developing working practices. Quality management is about ensuring the quality of the end product and fulfilling the customer's needs. Efficient working practices are an important component in the process. The objective of this study was to find and develop such practices for the team in question.

Software projects within the organization are executed with traditional project management methods. The team utilizes agile and scrum practices. New projects are typically built on top of previous ones so they are very similar in nature repeating the same process. Therefore process development and management set the context for practice improvements as well.

This case study was based on action research methodology, with observations done by the author. For each developed practice its case specific implementation was described and usefulness analysed.

As a result of this study, new practices for various purposes and different phases of project life cycle were identified. Improving quality is a continuous task; new practices still need iterations and improvements to better fit their purpose, and new ones should be developed in both team and organizational levels.

Key words: quality management, process improvement, software development, agile software development

TABLE OF CONTENTS

1 INTRODUCTION

In a newly established company the start-up phase focus on getting something done in any possible way that satisfy the customer. But soon after there is going to be a desire or need for reducing randomness in work activities, use resources effectively, and making sure that both employees and customer are satisfied. This thesis is a qualitative case study in a software development team where practices are still forming. Desire of both manager and team is to create together new methods and practices in this working environment to build quality.

Evolving team consists of professionals, with several years of experience in software development. Agile approach is used for software projects, and most project work can be defined as repeated process. Experience, project and process management form the base for action research, for trying new practices and making observations of how and why they fit in this environment.

Most methods cannot be copied exactly from neither past experience nor from theories, and need to be adjusted for the company needs based on employee skills, customer interaction model and work environment among many other things. The primary object of this study is to find and apply new practices for this team, to improve quality. As this team needed to “invent” these practices, the observations of this study work as an example and also basis for developing them further. The backgrounds, theories and observations serve for secondary objective, to provide examples and ideas for other teams in similar situation. Details of used data, software developed, people and company are excluded for confidentiality reasons, and consider unnecessary to fulfil the objectives.

Quality management is the main driver, with the perspective that work process improvement has direct impact on end product quality, and that doing the right things at the right time is cost efficient. Quality is not only task of a testing team, nor some activity that quality managers drive for the company. Quality should be built in in all activities, and most cost effective is to impact in the early phases of products lifecycle, so improvements in development are important.

Process and quality improvements require long term commitment, with continuous improvements. And it is also a work to do in company level. This study is limited to work activities that development teams are accountable and can impact directly. This study is limited to fixed time of 4 months. The scheduling allows different project phases to be included as some project is being finalized and others planned and implemented. Only specific practices tried in this time and done in development team level are included in this study.

2 THEORETICAL BACKGROUND

2.1 Project Management Theory

Project management has decades of history in different industries, as well as software business. According to Pelin (2009, 20-21) the tools and methods are used so long time and extensively that management model is developed with strong experience, thus only the essential parts that really work remain in the models. So relying on project management theories can be considered as using the best known methods (Pelin 2009, 26).

Project by ISTQB definition is "a unique set of coordinated and controlled activities with start and finish dates undertaken to achieve an objective conforming to specific requirements, including the constraints of time, cost and resources" (ISTQB 2012, 34). When work is a project, it is about planning and steering, and using those effective methods created for this purpose (Pelin 2009, 26). Lifecycle (of a product) with different phases like start, planning, implementation, follow-up, release, in-markets maintenance and ending of support, can be also seen as definition of a project (Leppälä 2011, 174).

Being unique brings in the challenge of trying to predict the future, how the implementation will happen and what will become the end result. There are also challenges coming from the environment: project targets become tighter due international competition, complex organisations, global project development with teams scattered in multiple locations, managing information becoming more complex, and lack of supporting software. (Pelin 2009, 21.)

Project planning traditionally rely on defining the entire project at once, but nowadays especially in software projects also content and completion criteria change during project lifetime. Therefore change management and/or the kind of flexibility agile working model introduces is necessary. Risk analysis need to consider possibility of customer requirements changing during the project. Especially in software industry competition and new technologies can change rapidly.

Cobo, Ortiz and Mataix (2010) have done research about "project and programme management" in multicultural development projects in different industries. The list of competences needed to succeed - to handle the challenges - is presented in table 1. It contains various types of skills and aspects, including taking care of quality in large as well as results oriented approach and enabling continuous improvement.

TABLE 1. Most important competences for programme and project managers. (Cobo et al., 2010).

Technical Competences	*Behavioural Competences*	*Contextual Competences*
- Parties involved - Risk and opportunity - Quality - Project Organisation - Teamwork - Project coordination - Scope and deliverables - Time and project stages - Resources - Cost and financing - Supplies and contracts - Changes - Control and reports - Information and documentation - Project Completion	- Leadership - Commitment and motivation - Creativity - Results-oriented - Efficiency - Consultation - Negotiation - Conflicts and crisis - Reliability - Ethics	- Project-oriented - Programme-oriented - Continuous improvement of projects and programmes - Coordination with the parties involved - Personnel management - Safety, health and environment

Project progress follow-up is important: any changes (usually delays) in schedule must be noticed and taken care of to ensure the planned completion date can be achieved without overloading resources in the end. Project planning includes breaking down the work in smaller tasks, and putting them in the timeline with dependencies. The splitting helps in scheduling, making the correct resources available in correct phase. This arrangement is called Work Breakdown Structure (ISTQB 2012, 50).

Figure 1 describes simplified view of project phases: planning, execution phases and closing (Project Lifecycle and Project Phases, 2009). Execution phases where development and testing occur in software project have great possibilities to add value in content and quality. Mistakes and quality problems causing delays may cause changes having great impact on final cost. Working in overload in a hurry is a very common reason to introduce errors in development phase. Not being able to notice the errors early enough impacts the end product quality and or project schedule. One of the purposes of testing is to provide information of project maturity, against which the schedule can be reviewed also.

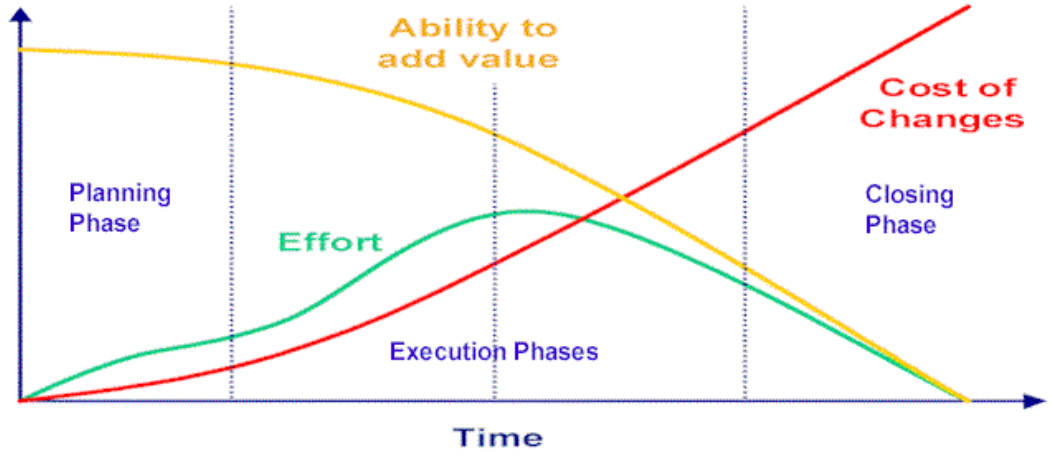

FIGURE 1: Typical project lifecycle (Project Lifecycle and Project Phases, 2009)

There are several studies and books written also about project failures, as quite often the end result is not what is expected, and/or time and cost targets are exceeded. Pelin (2009, 39) lists some causes for failing a project: just naming everything as a project without knowing what it is, no knowledge how to use project steering methods, planning and follow-up not organized, no systematic guideline and everyone working as they like the best, unclear target and expanding of content, forgetting of risk analysis.

2.2 Agile Project Management Theory with Scrum Practices

Agile is a working model that emphasize adapting to changes quickly, getting feedback early to do the changes, and people being in key role to interact closely with each other.

Agile manifesto has twelve principles:

- "Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Close, daily cooperation between business people and developers
- Face-to-face conversation is the best form of communication (co-location)
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Self-organizing teams
- Regular adaptation to changing circumstances"

Scrum is a working model that relies on agile principles and provides roles and practices. Scrum team must be small enough to interact with each other effectively. Scrum team has one *scrum master* that is not the leader in anyway, but facilitate meetings and supports the team. Top-down hierarchical management should be forgotten in things where team can take the responsibility, especially in scrum master role.

There is short meeting every day for everyone to brief what they are working on and if there is any trouble or impediment others can help with, and relevant people can discuss further and more deep outside the scrum team daily. Iterations are called *sprints*. Regular meetings for the scrum team include *sprint review* to see what was achieved, *sprint retrospective* to evaluate and improve, and *sprint planning* to decide what tasks to commit to. Daily and sprint meetings should also include estimating and following the time needed for getting the tasks done. For these regular practices some people tend to follow exactly the guidelines and durations, other perspective is to also adjust the practices also to be agile and flexible.

The tasks are small entities that are short enough for one sprint. Tasks are split from generalized items, user stories and epics. Project manager or product owner is responsible of keeping the bigger items in priority order, and relevant people get together to form the tasks before sprint planning. The planned work is called a *backlog:* scrum team handles with *sprint backlog*, and top level is a *product backlog*.

In traditional projects the approach is to agree the price, schedule, and content all at once, and many projects fail partially in one or more areas. In agile projects only schedule and price is agreed, but the content is modified during the project so that the result will still satisfy customers' needs better than in traditional projects. This is achieved by iterative approach, including changes in requirements management of product backlog and working close together with customer.

Traditional software projects can be planned to have several months of code development followed by testing, which often leads to surprises of not meeting the quality in time. But in agile projects also the testing is part of the smallest task: user story is not done before it is tested and working. In agile projects the most important content is done and completed, whereas traditional projects can meet the final deadlines before anything is fully working.

Traditional project management guides that all content must be decided in the beginning of the project with very detail task estimates in place, with addition of separate change management handling. Agile is guiding almost the opposite, not to go into too much details until very close, when it is known that this specific piece of work is needed. Agile projects have more flexibility and the change management is included in the regular practice. Backlog also contains items that are not necessary for the end result, new ideas and improvements are added during the project, and if priority is seen high enough, those ideas are worked with to have the details. Requirements priority is actively updated, like seen in figure 2.

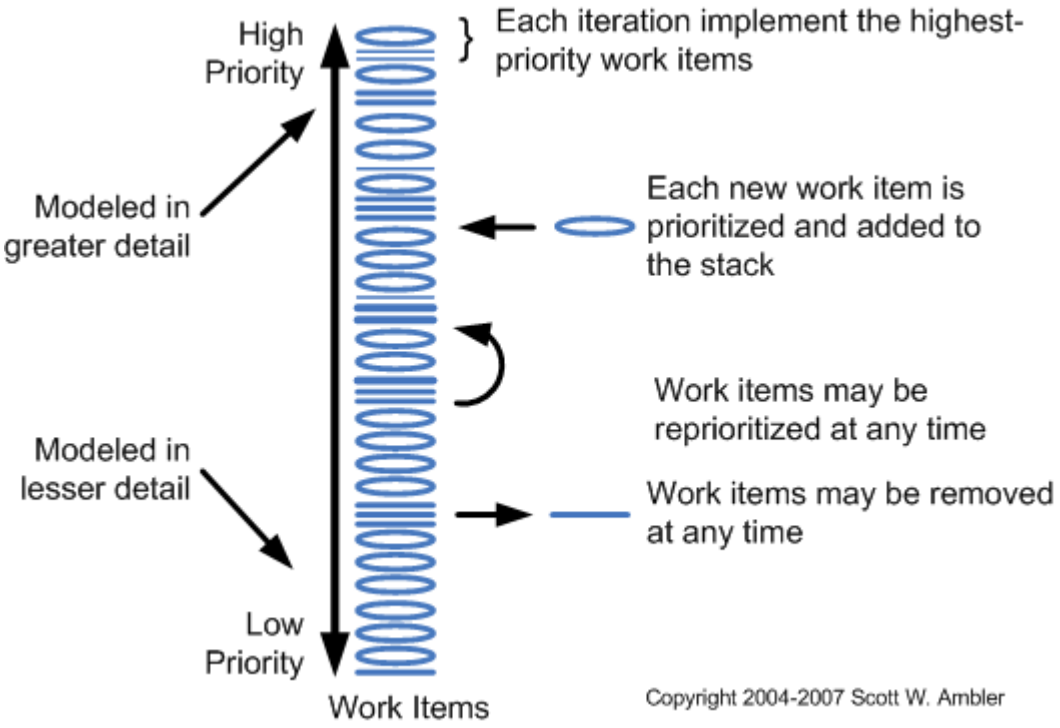

FIGURE 2: Requirements management process in Agile project (Ambler 2009)

The prioritisation work should include customer perspective in the first place. Presenting the developed software functionality and giving all-hands demos to stakeholders/customers frequently serves two purposes: to collect frequent and direct feedback easily, and customer can gain confidence in project going to correct direction.

Ambler's study of agile success factors presents that with agile teams are able to deliver better quality, more of the required functionality and improved economics. (Ambler 2009) Comparison is shown in figure 3. Part of the improvement is about short feedback cycle and better interaction with customer, but also it is about agile teams are working smarter, but not harder.

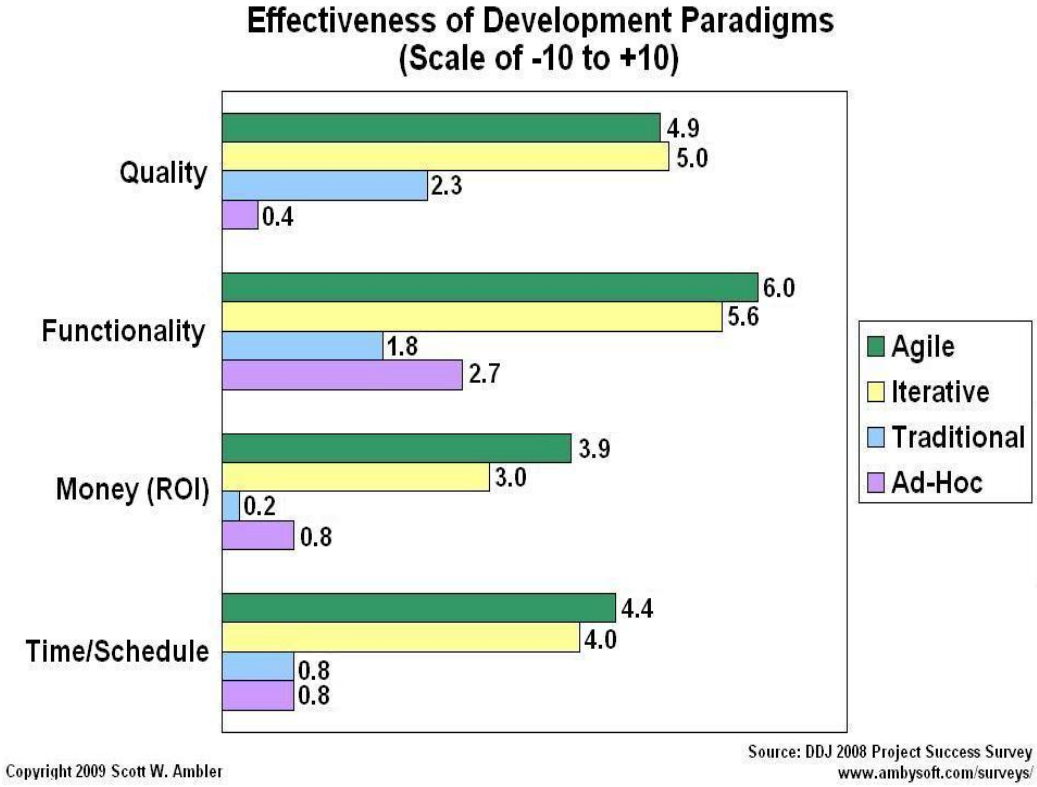

FIGURE 3: Ambler's comparison of agile development strengths (Ambler 2009)

Monitoring changes in conditions, and adopting working methods accordingly is considered as a principle to effectively improve both projects and processes (Leppälä 2011, 175). Mechanism of adopting, small updates time-to-time, over generations, is known as evolution (Leppälä 2011, 175).

In human activity evolution is born from the practice, from the needs of improving organisations and technology by minor yet continuous improvements. As conscious mechanism it is fairly new ideology (Leppälä 2011, 176), but the mechanism matches with continuous improvement and short cycle iterations of agile theory.

Time management of daily practices, too, is important part of scrum practices. Meetings have fixed duration, for example daily meetings should not last longer than 15 minutes

and other meetings are time-boxed for 2-4 hours. Iterative mode of projects is also considered important for practices as the retrospectives purpose is to enable continuous improvement, collecting feedback in the scrum team and agree on changes.

2.3 Process Management Theory

Process and project are not to be mixed with each other. Process can be used to define the models and improving functions of organisation (Pelin 2009, 22). A project only occurs once, but when similar type of customer projects, consisting of same or similar phases and roles, it is possible to define processes to cover project development. Process management emphasize working effectively, produce consistent quality and the importance of continuous improvement.

Processes are way to standardize the repeated work practices, to save effort when work phases and roles are agreed beforehand, avoiding confusions and spending time trying to figure out what should happen next. The main functions and key processes providing services or products to customers should be known or descriptions should be available for everyone taking part of the workflow, allowing seamless collaboration between teams and working towards the same target. According to Laamanen (2009, 21) process planning should start from and end from customer action, to help understand what is really critical to achieve good results.

Process work model should provide support for people already capable of doing their job on how to do correct things in practice. Support can be given as documented model, checklist or form. And it needs to be kept simple to be practical and frequently used. (Laamanen 2009, 37.)

Process can be described in written format, but there are several types of visual models that can make it easier to understand. The pictures are not the purpose, but a communication method and means to model the functions of organisation in order to understand, analyse and improve (Laamanen 2009, 75).

Because process is repeated, its performance can be monitored and measured, which provides the possibility notice deviations and to improve (Laamanen 2009, 20, 152). Deviations can be acceptable exceptions, reason to re-define the process, or cause for quality problem. Laamanen (2009, 150) states that numeric data should be measured and the measurements can be used to give a glimpse of reality. (And same thought is formed decades earlier by Peter Drucker: "If you can't measure it, you can't improve it".)

With deep understanding of the process and careful analysis also the numeric measurements can be interpreted as actions to change and improve. Numeric data can be used as indicators to communicate and follow-up progress. But without the understanding people usually tend to make decisions that work against the real goals, against the company strategy and the customer need (Laamanen 2009, 150-151, 205). Numeric data does not form information or provide actions to change without effort of learning and analysing. Figure 4 represents a control chart form of process measurement, representing the three phases of Juran trilogy (ASQ 2013): quality planning, quality control and quality improvement.

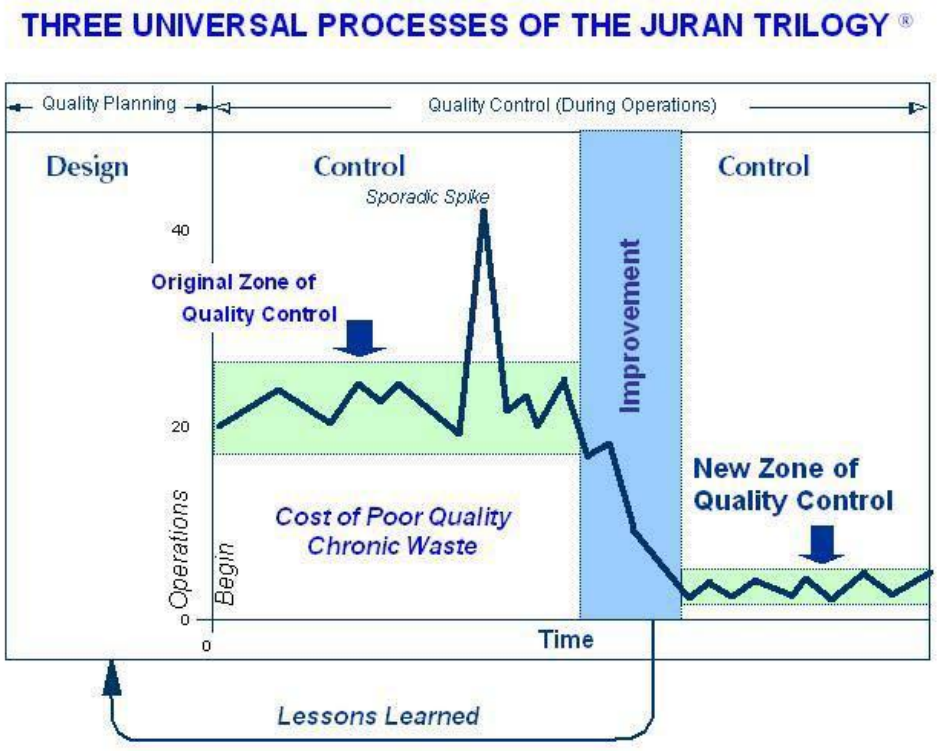

FIGURE 4: Quality improvement can be introduced over time by measuring and monitoring process operations, evaluating and redesigning the process. (MSI 2011)

Since improving performance and quality of process is important, the theory includes methods and guidelines to do it. Sources for improvement needs can rise from customer reclamations, errors in products, deviations in process, audits, employee complaints or employee improvement ideas. (Laamanen 2009, 191.)

General approach is to not set the entire process under evaluation, but trying to remove a small identified problem at a time. Change is to be done and analysed iteratively. The improvement cycle with iterations going through four (Plan, Do, Check, Act) or five steps (Define, Plan, Execute, Check, Learn and generalize). Figure 5 shows how Ishikawa splits the cycle of four into six steps. The cycles are introduced and used by many of the leading authors of process and quality management theories, as well as those who promote learning by doing and reflecting. Laamanen mentions David Kolb and Reg W. Revans. Other authors include Kaoru Ishikawa and W. Edwards Deming. (Laamanen 2009, 191, 211; SkyMark Corporation, 2013.)

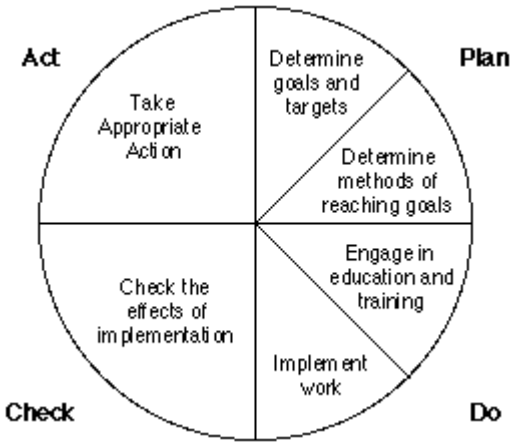

FIGURE 5: Improvement (or learning) cycle defined by Ishikawa (SkyMark Corporation, 2013).

Choosing problem solving methods is subjective to the problem type, like simple temporary issues and complex recurrent issues, and whether there is quantitative or qualitative material available. Laamanen (2009, 214-216) lists graphical quality tools like flow chart, control chart, cause-effect diagram (also known as Ishikawa and fishbone diagram) for quantitative material. There are also quality tools for grouping and prioritizing more complex data, evaluating risk and impacts. He also mentions comparison methods, like group work and benchmarking.

Process management can be seen as a method to prevent creativity and cause dissatisfaction for employees (Leppälä 2011, 174). Laamanen (2009, 44) states the process management also targets to include innovation. Process needs to evolve from chaos to repeated functions that can be modelled and measured, to mature from reactive to proactive by including analysing, and finally include innovation and creative methods to achieve world class.

Process management does not work usually because of poor descriptions lacking roles, responsibilities, management or customer, or on the other hand because descriptions are too detailed and complicated to digest and take into use. If descriptions exist, they are not made available to people who should be using them, the usage is not being monitored in any way, or simply there isn't good enough tools or knowledge to implement the process. It is also common to use the word process for just about any trivial habit without understanding the full purpose of process management, which often leads to undermining and resistance of process improvement proposals. (Laamanen 2009, 297.)

2.4 Quality Management Theory

Quality is defined as "the degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations" (ISTQB 2012, 33). Quality management coordinates and directs the activities to achieve quality. It usually includes the establishment of the quality policy and quality objectives, containing quality planning, quality control, quality assurance and quality improvement. (ISTQB 2012, 33)

In software projects there can be three different quality targets: end product quality, process quality (working methods, schedules, costs), or service quality as in the experience of participating the project (Lehtimäki 2006, 67-68). The contents of these targets should be defined though, as there is no reason to build the kind of excellence that customer is not ready to pay for. For example purchasing a custom program to be used by two users should work correctly for the purpose, but if ensuring it will work for thousand users or spending time in fixing non-functional cosmetic errors increase the cost, it is probably not necessary and not in interest of the paying customer.

Phillips (2005) mentions that not only the product quality but also the quality of process making the product/service are the essential to project manager, because the process quality has direct impact on product quality. The better management on work process, the higher the product quality (Phillips 2005, 320, 326.)

Based on his working experience in multiple software projects Lehtimäki (2006, 67) describes three levels of quality maturity in organisations:

0. Organisation has no idea about quality, functions vary in many ways.
1. A separate quality organisation, quality managers, quality handbook (ISO 900)
2. Quality is integrated in all activities.

According to Lehtimäki (2006, 67) the level one is not sufficient, since separate organisation can give the false assumption that not everyone is accountable of the work quality.

Quality management activities need to target in full lifecycle of a software project: from receiving of purchase request all the way to delivering the product and getting it into markets. In quality management, it is highlighted quality is not only about testing, but something to be built-in in everything.

In development phase of project an example of quality violation could be for example an internal testing tool becoming unreliable, and nobody escalating the issue to be fixed to save resources and enable software test results reliable; unreliability causing the fact that most developers stop using such tool and try re-inventing other methods to validate their changes and testers re-executing the program several times to increase the probability of results being correct. Failure in detecting the quality issues does not always become visible to customer, however it still can introduce a problem with resourcing and increasing the risk of some activity for the software and customer request is not getting fulfilled.

Analysis of identifying and prioritizing risk would be also needed in planning phase of big projects with new technologies or complex changes; the greater the risk, the bigger the impacts of quality failure causing trouble with both schedule and cost issues.

Validation is about confirming the requirements are met. During lifecycle of software project the cost of error fix is multiplying in every step of the way. Black (2007, 111) declares in figure 6 it is 1000 times more expensive to fix a bug (=error) in released system than if it was handled already in requirements phase.

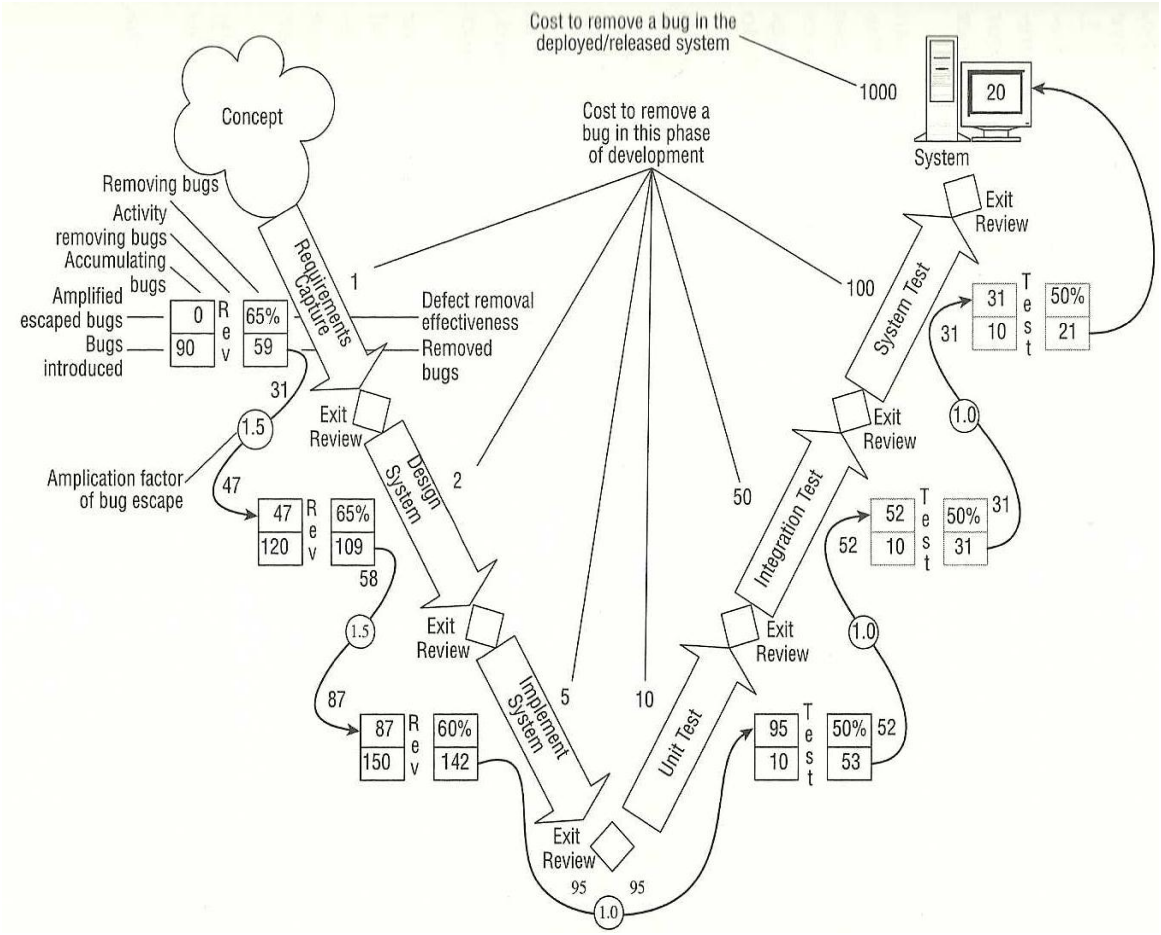

FIGURE 6: Black's (2007, 111) calculation of bug removal cost in different phase of software project.

Laamanen also say it is cheapest to do everything correctly on first time, it has been noticed in process management as well (2009, 163). Myllymäki (2010, 161) adds that it is not only about the cost: the more time is spent on definitions and review in requirements/planning phase, the better commitment is achieved for the whole project. Total cost of quality needs more details though, as activities can be split into prevention, appraisal, internal failure and external failure costs (ISTQB 2012, 15). Smart use of resources and activities needs to be monitored in all phases of project, to build a business.

3 COMPANY ENVIRONMENTAL BACKGROUND

3.1 Development Team in Case Study

Company has been established for decades, and even with current challenging economic times it has been profitable and growing. New branch is established for software business that is expected to grow in future. This branch is referred here as *organisation*. The office where this thesis work takes place has been established about two years ago. First projects have been successful, and there is growth in all areas: more customers, more projects and more employees. So the team and the organisation are still in evolving phase. There is expectation and demand of this organisation to become efficient and produce high quality software releases to actually reach the promise of growth in future. Investments done for the start-up need to be changed as results.

Organisation consists of several teams. This team in focus of the thesis is a software *development team*, consisting of two managers, three *scrum teams* of developers writing the code and one *quality assurance team*, with total headcount of 30. Writer of this thesis is part of the *quality assurance (QA) team* that handles requirement management, releasing activities and various tasks supporting the development team.

In the organisation there is a sibling development team in another location, doing similar work but with different projects. It also contains the *testing team* that is shared with these two. There are also few other software development teams in close collaboration, responsible of related software components. All these collaborative teams are referred as *domain* in this thesis. Figure 7 illustrates the development team and relations of the company doing software releases.

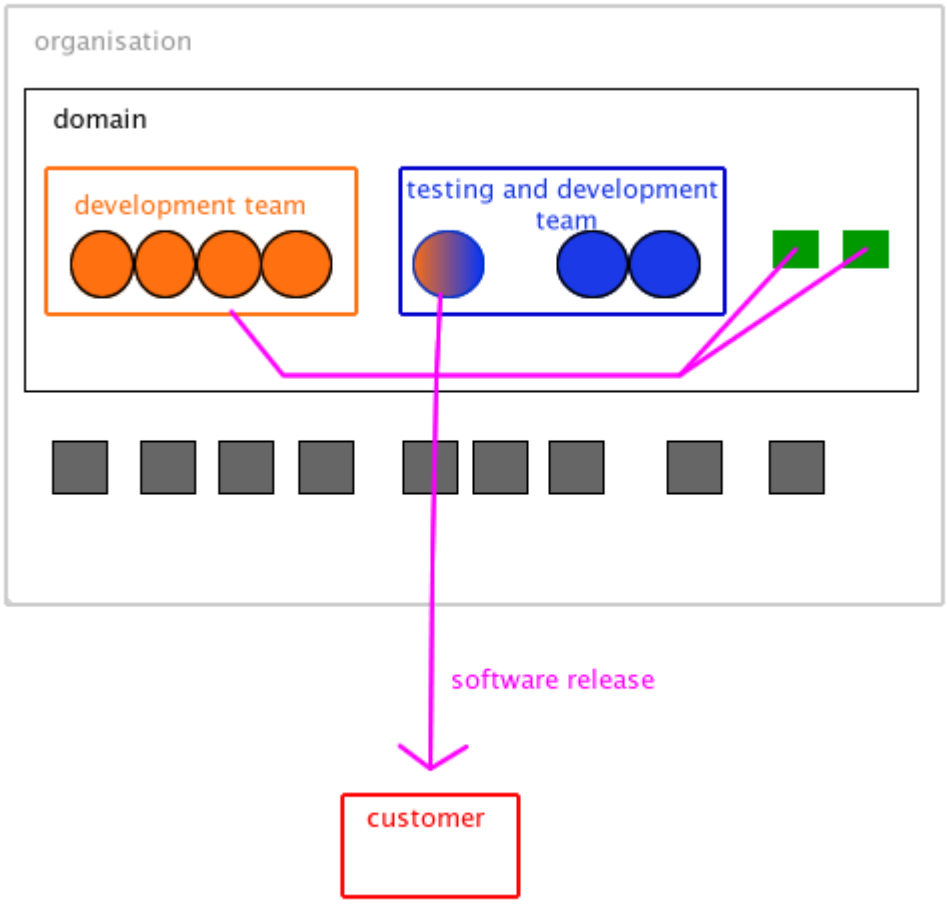

FIGURE 7: Development team consists of 4 small teams: 3 scrum teams and QA team. Testing team is shared with another development team. Other teams in same domain and in same organisation contribute to produce software releases for customers.

It is worth noticing that in the beginning of this study there are newly employed people in the development team. The split to scrum teams and QA team is relatively fresh change, as before whole team used to be just one scrum team. It is expected advantage that newly employed people give fresh perspective, as they do not know or care the challenges in the very beginning, and therefore can raise the expectation to level of more stabilised company. It is also a challenge in two ways: The current status needs to be understood, before it is possible to study how to change it. Some people do not have insight of what the current practices and processes are, or what are the problems with them. In start of this study there is a lot of relying on second hand information, which can also vary depending if the information is shared by person who joined six months or two years ago.

The employees in the team are all experts in their areas of knowledge, with around 10 to 20 years of working experience in the business area in various companies. So as the office and resources in place, it has been possible to drive through the first projects with hard work on top of the experience. In this type of software industry the ways of working in any size of a company are similar in many ways, and most people are familiar with projects and process managements, as well as modern software development technologies like agile and scrum. With growth, however, there is increasing need to set focus on quality management together with resourcing. Working harder or increasing team size forever is not an option, but working smarter is: focusing of doing the right things at right time. Also purpose is not forcing some old habits without studying if and how they add value, and are applicable in current environment.

Customer here means an external and paying company for which the organisation creates and offers software releases. *End user* satisfaction is an important quality factor to consider, but in business perspective it is further away, being a customer of a customer. From the team perspective also the organisation or other domains are *internal customers*. For clarity reasons the internal environment is referred generally as *stakeholders*. Quality criteria need to satisfy both customers and stakeholders.

3.2 Present Practices of the Team

In this thesis a *software release* done for a specific customer or created as offering is considered a project. Software releases contain some similarities with previous releases, existing functionality to be delivered with different configuration. Often some new and improved features and functionality are included in addition. A *new feature* development can be also considered as a smaller scale project, or subproject inside a software release.

Projects are somewhat combination of traditional and agile projects. In organisational level there are no guidelines or practices to use agile. Both the schedule and content must be fixed and agreed beforehand with all stakeholders, so that entire organisation - other dependent development teams, system integration and system testing can also

prepare their tasks and duties. Yet sometimes changes are needed fast, and the team must be flexible.

Team uses scrum practices in some ways that are found applicable, and has done that from the beginning. Daily meetings are kept in scrum teams and considered important. Planning is organized to run in sprints and iterations. But for example backlog is run in various tools and there is no combined way to see the prioritization and progress. Sprint retrospectives are allowed in scrum teams, but rarely produce actions to the entire development team. There is no regular practice of project (or iteration) retrospective. Practices to tie together the three scrum teams and QA team activities are also forming, as the split is recent.

Software release projects are the key business, and there are several projects on-going in different phases. The phases of the projects are defined and milestone quality criteria for each phase are determined. Each team contributing to a project basically knows the expectations. There are many practices in company and team level that are referred as "this is the process". But the milestone criteria or project process descriptions are not easily available or recognized by all employees. Also project follow-up is not shared effectively to teams, but amongst management. The information flow is hierarchical. Project priorities and progress are being shared by the manager of this domain to development teams in meetings. Employees are frequently asking about both, to gain confidence the sprint planning is done correctly. In some way this information practice seems to be reactive rather than proactive.

There is little or no requests from development team to have more detailed process descriptions. Perhaps there are previous experiences of such documentation not being helpful or applicable, not making the work easier thus not adding value. Agile approach that employees are accustomed with also emphasize documentation and bureaucracy should be avoided, so even with justified reason there can be sometimes hesitation or even resistance against more documents. Still the attitude is about wanting to do the work right in the first place.

Customers or business people are usually too far away from development teams to really get involved frequently. And many counterparts and stakeholders cannot be met fre-

quently face-to face. There are no practices, like demo sessions or hands-on presentations of the features and software quality, that would serve the purpose of possible customer feedback.

Testing team is not directly part of the scrum activities, as it is independent group, and not co-located. Information between teams is shared via emails, and call conferences between few persons with key roles in both teams. With QA team formed, it is expected the collaboration and practices between development team and testing team will improve. QA team responsibilities are not limited to the testing aspect, but also include for example requirements handling and releasing activities. As scrum teams task list contains concrete items of developing software, QA team task list contains more abstract items like improve releasing process. There are known problems given by manager to start with. Team is expected to proactively find sources and root-causes to problems, to drive activities improving process performance and quality.

Currently team and organisation do not fit directly anywhere in Lehtimäki's view on quality maturation levels: Zero level maybe applies to some things, as the quality practices are not being focused. For level one there is QA team nominated to handle the issues, but there is no organisation wide network of people doing the same. There is some quality criteria specified considering software delivery and testing. The most mature quality level is the team's long term target. And some built-in quality practices already exist, such as developer testing and peer review being required before submitting changes.

3.3 Detailed Objectives of the Study

Every professional test engineer knows to file an error report when they find something wrong in the software under testing. It is their responsibility. Developers writing and reviewing code know to follow code conventions and architecture design, and fix or address any critical violations. Both of those activities impact directly on product quality, and activities and monitoring are needed to confirm the practises are really used and working. But when there is something affecting the work quality, it isn't always so obvious that the issues get addressed and fixed. Quality management tools, methods and

practices are needed to handle working practices that will ensure quality of the end product. Service quality is excluded from the scope of this thesis, as customer is not directly involved.

Team is motivated to do the job and to improve practices. Desire is to work more efficiently, improve planning and execution, and build in quality in every phase of projects. Quality management defines the direction of this study: to get quality built-in with least effort. Evolving environment provides the positive attitude. Learning of team work and existence as one team is on-going, and therefore it would be good to have methods and practices to also involve entire team to work together. There are probably also many connections to other teams that are not yet strongly established. Main commitment for everyone is naturally in doing the job, so any practice changes cannot take too much of working time, and changes need to be justified and make sense.

The type of improvements should support skilled people and the team in whole to succeed. The desire is to continue working with respect to the agile principles. It is also expected that ideas and practices coming from inside the team are valued, because then team is motivated to keep them and develop further.

None of the theories are applicable alone, but target is to find good combinations that are suitable for this team, in this company, this environment and working conditions. “Be agile about agile” is a suitable quote, of which origin is unknown, but it does summarize pretty well the relation of known models and how they should be used in different work environments: To know and understand the theories behind, and to try and select the best ways that work well, benefitting the employees, employer and the customers.

The study includes activities that the development team can impact directly. Great deal of the project start-up phase is excluded, as it involves managers and architects contribution, but very little impact of development team. When the set of features, customer needs and configurations are decided, then development team is starting the planning, implementation and testing activities. Also most of the project work is done to achieve certain milestone that can be described as “ready for customer delivery”, thus the

maintenance, in-markets and final closing of the project is excluded from this thesis scope.

The study is limited to duration of four months, with agile approach of using a fixed time. Duration also provides an example of how much or little changes and improvements can be done, in addition to performing the regular work. The expectation is that good practices in correct place can be adopted quickly, and modified over time to fit well. Limiting the duration also includes the fact, that it is not possible to try out all methods and practices that are seen important, and therefore this study is not presenting a view of all the things this or any other team should have to solve a quality problem.

Measuring the process efficiency and software product quality is important, but excluded from the scope of this thesis. Mainly because the scope is defined as direct impact of the development team, and projects related metrics is an organisational issue. It would be also quite a big work to do and does not fit in the schedule of this study, since the measurement system does not exist in a form that would show results in this context.

This study describes some of the changes to working practices, that involve the quality assurance team, and all or part of the development team. The ideas and inputs are also collected from inside the team, by one person or after discussions with many. The key information is the observations followed by a summary. Results should provide insight how to continue using the practices improvements, and if same or similar changes or approach could be applicable in different environment for readers. Some ideas that are suggested but not tried are therefore excluded from the study. There are no changes presented, that are given fully from outside the team, like actions organisation is expecting all teams to do. Also, there are no changes described that involve only team specific details, like actual tools or specific roles.

4 METHODS

4.1 Action Research

Action research is a form of qualitative research in which an intervention is brought to community, situation is observed and evaluated. Introduced change forms a study when community and researcher try to find solution and working together to achieve a common target. Researcher role can vary from pure observing, to being a full member of the community. (Eskola & Suoranta, 128-131.)

Role of facilitator in qualitative research method is essential. There is more freedom compared to quantitative methods in planning and implementation of the study. Empirical and experimental approach also requires trying of new things, and searching materials to form new ideas, viewpoints and hypothesis, rather than verify and proof existing ones. Conducting a quantitative study can be argued as non-scientific, as it cannot be reproduced, and rely strongly on common sense and previous experiences of facilitator used in a unique case. (Eskola & Suoranta, 20-21.)

Challenges are to identify hidden personal pre-assumptions resulting choices, and avoiding driving a personal goal rather than the common goal. Other ethical questions also need to be reviewed, like what is the impact of the facilitator's actions when being part of the team to study. (Eskola & Suoranta, 20-21, 52-53.)

Narrative descriptions can be also part of the study, building logical structure that is natural to understand, and can be analysed further (Eskola & Suoranta, 22-24.)

4.2 Research Plan

This study is fully empirical and qualitative research, conducted in environment where several management theories are applicable. The employer recommendation to development team is to come up with good practices how to work smarter, and share those "best known methods" inside the organisation. But it is not intended in this short time

the practices to become permanent and finalized in way that can be said they are “best known methods”, and it is also not included that the practices are shared yet. Performing the research chapter includes that the trial and error phase is described in a narrative way that presents the planning (setup), execution and observations. This part provides a variety of written information based on which reader can evaluate if and how they can use it as an example, without actually taking part or seeing how the practice is used. New improved practices are summarized to provide an overview of the results that apply in this case.

Doing this kind of improvement together as a team is a form of action research. Writer is full member of the team, and is facilitator and observer of the practices included in this study. Using the new practice does not require full team to participate. Team working together also means that the ideas are shared both formally and informally in various situations of daily work. Most of the development team is not informed the work is used in a study, and some members know about the study without knowing the exact scope. This is decided to exclude the possibility some actions are done for any other reason but helping the team to develop the practices.

It is not decided in advance what changes are introduced and how. Writer’s role in the team allows decisions and actions taken directly, but the working model - like regular meetings with quality assurance team - includes the plans are shared and discussed, minimizing the impact of personal pre-assumptions and introducing big changes without planning together in advance.

It is not necessary to include exact details like employee names or forms of interview, since it is not intended that the study can be reproduced. Instead descriptions and observations are relevant part of this study, followed by summary and analysis. Common sense and previous experiences are the backbone of the study, but in addition the theoretical background and existing information about common problems are taken in advantage of.

5 PERFORMING THE RESEARCH

5.1 Improving the Project Planning Phase

5.1.1 Creating a Project Test Plan Checklist

To start with the improvements, some top level items were given as assignments to QA team to handle. One of the tasks was to come up with a test plan for the next new project that was soon approaching the implementation phase.

Few initial assumptions were made. First, there is no existing practice for project test planning. There have been projects before, and testing has been done as the projects are finalized, but something must be missing in either planning or execution, since a test plan is being requested. Second, there are other projects to follow, so the plan should be reusable. So this task could contain both a template to be used for any project, and some detailed form of test plan specific to the starting project.

Like process management guides, the current practice needs to be understood for the improvement. As some team members were newly hired to the company and team, it was in all ways natural to start investigating what are the current practices, and what is the missing part in testing that this plan needs to solve.

The task started with informal interviews and discussions. The members of QA team asked questions, held meetings and interviewed colleagues to understand what the quality issues that should be solved are. Weekly meetings were held to follow the progress of this and all other tasks, in similar manner with scrum team practice but combining the daily and sprint meetings purpose in one. Online meetings were also held with testing team, to clarify current practices and quality criteria used. The results of discussions and meeting minutes were written down in a document, to track the progress.

With the given information there were two discoveries. The testing team already is working according to a plan they know, and that quality criteria exists and is used. It was not then the purpose to evaluate that further, since obviously it was not anything

new to be created, though it could be something not written down and shared to the development team. The second discovery was brought up from discussions about terminology: if testing team has established and working practice, then what it is in the *validation* that does not work. Validation is about confirming the requirements are met, so more questions were raised: what are the practices, where and how the project requirements are created and shared. At this point it was also noticed that nobody really defined the task target in detail to start with, so it would be rather hard to hit the goal. But since the new questions were important, and parts of the requirement handling would be also regular task for QA team, the investigation continued.

There was a lot of material already available from all the knowledge QA team members had and collected. What are the common things forgotten in planning, like legal issues or specific testing phases, or keeping the requirements list valid in all details and schedules. It was decided to then create a template: a checklist that covers all things to remember in project planning phase the problem was, there was already too much data considering it should be formed to a simple checklist to use easily.

The work seemed to stop, and there was no pressure from team manager to come up with solution. But it was kept as an open task, to follow the progress in weekly meetings. One day before a weekly meeting, one member took the data and started reading it through. All of it seemed relevant and important, but not all the things were actually required in project planning since part of the things would handle more detailed the features inside the project or just common practices that issues that were not right at the moment. The data could be split in three categories:

- Project level - needed rarely, mainly tasks to management.
- Feature level - needed frequently, mainly tasks to implementation and testing
- Daily work level - things that seem acute and needed to solve once.

The data was formed into two checklists: one for project planning and another for feature planning. The daily work level was left out from these two documents, but kept as open new things QA team would be responsible to continue working with.
The next question was, if we get the team to agree with us, about the contents of the templates, and the practicalities how to make it work. The checklist did not provide a process view of who does what, how and when, but more like items someone has to

remember. Expectation was that template would have usable and valid information, but there was not enough confidence to start pushing it through. These documents were not published to ask review from neither of the development teams nor testing team. The task was also to create specific plan for a named project, which still did not exist.

By the project start, the decision was made for QA team to try out the project checklist, and see if it needed adjustments. It helped the QA team members in some duties. But the start phase was surprisingly quick, and there was no specific project plan created that would have had need to follow during or after the project. The implementation and testing started mostly in same way it had been done before. With project on-going the planning task was waived, as then it was time to learn in practice how to deal with all the phases.

Open question still was the purpose of test plan: should it focus on the functionality and differences of this project compared to previous, should it define strategy of what and how to test in each level from development to end user testing, or something else.

Observations are that this task kind of failed, as the original goal was not set and the preconditions were not identified to begin with. And perhaps some assumptions were made, without making them visible and asking if they are valid or not. The classical reasons why improvement projects would fail were met: target cannot be reached when it is not defined in the first place; and planning a change requires understanding the starting conditions first.

There was progress, and mainly the difficult task got pushed forward with the help of weekly follow-up, even though it was done by the team itself. Work was done with small number of people inside two teams, with discussions and documenting. Benchmarking to other teams practices, or involving entire development team or properly informing them was not done. But there were results found out around the topic, some basic understanding of required steps in checklist, and good background information to continue with several more improvements.

There was a large amount of improvement ideas and needs identified, by making conversations, writing down questions, answers and meeting minutes. To help processing

the qualitative material, a classification method was used and ideas were handled in three levels (project, feature development, daily work). Even though some of the material collected was not helping to resolve the testing plan problem, it could be re-used in continuing improvements.

5.2 Improving the Project Execution Phase

5.2.1 Regular Project Status Sharing in Development Teams

There was already a project progress reporting practice existing. However, the metrics and insights document was created for management meeting, and its purpose was for all teams to report progress to project management. Teams in the domain were informed, but not in regular systematic manner. It was noticed quite often questions like “are we busy, or busy-busy” were made, trying to understand if the project is progressing as expected, or if there are any delays or extra effort needed. Everyone working in many projects, it was anticipated that towards the end there could be huge workload ahead, if not keeping the focus on right things.

While the task of collecting the weekly material was given as regular task for QA team to handle, it was also decided to start sharing the progress report by email directly to the entire domain. Since the domain contains several teams, and not all teams work with the same project the worry was that many people not involved would feel regular weekly email excessive. To address that worry, the email subject was designed to include keywords and project name, so anyone can easily create a filter in their mailbox.

The email would contain metrics, like error count and test execution results. With milestones approaching, the targets were also included in advance to focus. Some written analysis was included when there was a change unexpected, or something special to focus or highlight. While the purpose is to show that the project is not yet ready, and often the highlight is pointed at issues that are behind from the targets, also smaller improvements were highlighted even though it would be only applicable for that week. If all the report is just negative, it might not get the desired attention.

Surprisingly, there were no complaints of too much emails coming. But it was easy to reply to the email, and suggest small changes and improvements. Some comments were given this was good information, but with question if the progress is on par with other teams, so to fulfil this request a bigger set of data was included every now and then, still keeping the regular email smaller. On the other hand, when the project final milestone was coming closer, the frequency changed to daily email with only the issues preventing the target, but still keeping the bigger summary email once a week. Some feedback was given that this emailing of the status would be a practice to keep. It was like the lack of information was really noticed only after it was made available.

Observations: The decision to share the report was based on the theories, like project communications and agile principles of involving the team. Also some of the information would have been easily available for anyone, but when the way of collecting the data was made standard way, it would really provide a reliable measure to follow the progress. And the information collected and made available meant that scrum masters did not need to repeat the steps to prepare for their sprint planning, so some effort was saved.

From quality perspective it was not always a good thing to report exact numbers and the negative message “we are not there yet”, but luckily there were no identified issues where anyone would do the wrong thing or dirty hack just to get the numbers look better. Still, in many conversations in the hallways, some people tend to repeat the numbers in a manner that would require ensuring “it is not the end of the world, keep calm and carry on, do the best you can”. On the other hand, when scrum team specific error count was made visible, some scrum teams made it a bit of a competition and fun, even though reaching the target was serious business.

Emailing was used as the means to deliver information, as it would reach all employees better than hierarchical system of key persons discussing in meetings and delivering the information forward. The data collection was done from different tools, and it required some effort to format as message each time. Tools could provide the visibility better in company level, so that all projects and teams could have consistent views automatically as dashboards that would enable more pro-activity in development teams. Time saved from collecting the data could be used to make it into information.

5.2.2 Using Exploratory Testing for Maturity Evaluation

Quite often software project maturity measurement rely only on metrics provided by testing: pass rates of test execution, and amount of open errors. Especially close to project finalization the amount severe errors is a measure to follow daily. But the numbers do not always indicate the sense of maturity that one would get when using the software. Development team doing fixes one after another can get frustrated when the numbers don't progress well enough. Another source of frustration can be the sense of not having control of their own asset, as the evaluation is done testing teams with no direct contact and therefore outside of the community.

Project success challenges are also psychological. It is the people who are doing it, in every step. People need to understand the quality, feel the success or loss, and learn what is important and how to do it in the next project, working again with people. (Leppälä 2011, 174.)

The development team did not have any practice of doing regular testing on real environment - besides validating self-made individual fixes and changes of course - exploratory testing sessions were started. It is informal testing technique (ISTQB 2012, 21) that does not contain directions what to test, but participants trying out freely. When dealing with software that will be available for any type of people, anyone in the company is qualified as tester in this session, and also presenting the customer providing feedback.

From experience it was known fact that all people (developers, testers, managers) are individuals, and find different type of issues when executing manual testing. When one person finds an issue and explains it out loud, another person trying the same or similar use case can find a different problem. Multiple persons doing about the same also provides information about reproducibility quickly. It is essential to keep the exploratory testing session informal and discuss freely. Another element of informality is the scheduling: session duration can be defined to one hour during which participants can come and go freely, spending anything between 15 to 60 minutes. If meeting room facilities allow everyone is also allowed to bring a coffee cup and have a break during the session.

The purpose of Exploratory testing session is not to do just anything freely. The scope must be defined first, and the targets of what kind of results are expected. The target can vary a lot, depending of the project phase and software maturity. Target and maybe some hints on where and how to start can be needed, but with the discussion and findings it is often continued itself from there. Collecting of the results must be organized in some way: whether is a paper to fill for everyone, paper to circulate from one person to another, or one person working as secretary to write down findings from all.

First session was organized because of the complaints that the big error count would not tell the correct level of maturity. So a small number of features that have been prepared long time ago and small number of feature recently changed were selected as topic: For each feature on paper was provided with scale sad-happy for everyone to mark their opinion of maturity, like shown in figure 8. The evaluation request was expressed with question: "If you purchased this software, would you be happy?" In addition, papers were available to write down new errors.

The development team was invited, as well as people from other teams with less experience of the used software. The number of participants was small compared to the invitations, about 10 people. But even after short period of time, there was a list of new errors filling up, and people leaving the session with marking their opinion of the maturity: For many it was a surprise that actually the maturity was far from perfect, rough estimate of average grade from the pictures was around 7, if scale was 1-10. And not many people would care to stay for the full hour.

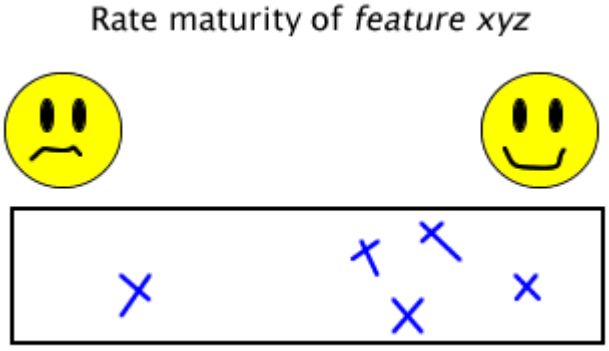

FIGURE 8: Maturity evaluation form with open scale from sad to happy.

This subjective grading, the happy/sad papers, were also put visible in team premises, so that those not joining would see it, too. The error list was also put visible. All found errors were included and reported, the reporting is not limited to the features being named as focus of the session.

With next sessions the maturity grading was left out, and the focus was on finding the critical errors, in any of features changed or impacted recently. Not all the time the errors were new, but the severity became clear as issues often occurred in short time. And none of the team members would be proud if customers would see it. Also as the error were written down during the session in detail enough to be searched for existing error reports and submitting as new ones during the same day.

Observations are that this type of sessions are working well for the team. It was anticipated that some errors will be found, but still there were occasions where severity or amount of new errors came as surprise, knowing that test automation and several phases of testing had to exist before - including the developer testing - that missed the issue. Big part of the regular code writing work is done alone in own desks, and the joined meetings are seen as time away from the actual work. These sessions represent something in between: working together. Even if customer is far away, this method does give a hint of what would it be like to give a demonstration frequently. The most important part seem to be the ownership of the quality, and the rather easy way of prioritizing own work. Some occasion a developer quit the session with comment “I have to leave now, because I’m going to fix this by tomorrow”.

5.2.3 Using the Fishbone Method for Finding Root-causes of Problems

Process management teaches to notice the exceptions when something unexpected happens. For the sake of process, it should be investigated in order to see if process is correct but people need support, or if process itself does need change and improvement. Agile reinforces flexibility to change, and trusting the professionals doing what is the best, so process type approach can be seen contradictory. But even so the quality issues that are caused by doing things “differently” should probably be investigated and ana-

lysed to find a way to prevent same thing happening again. The taken action can be lead from the quality issue, rather than the people.

A method known from process improvement is to ask “Why?” so many times, that the final answer reveals the root-cause, and makes it possible to find a working solution. The questions should not focus on one possible solution, but to include all the variety, like tools and equipment, people and practices for different roles. This root-cause analysis can be presented in visual format known as Ishikawa or Fishbone diagram (Silvers, 2004.) shown in Figure 9.

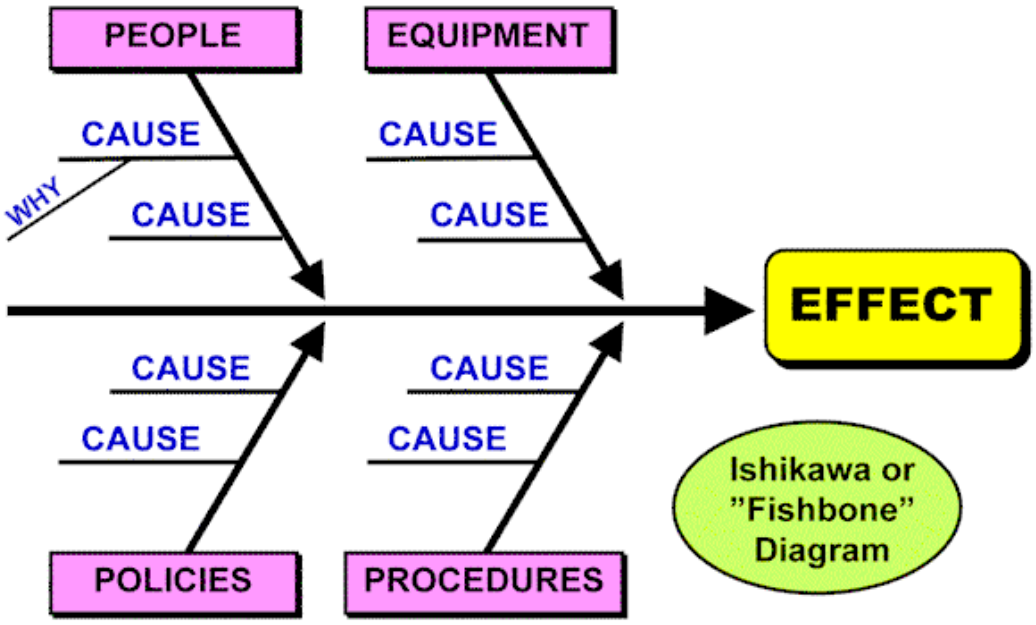

FIGURE 9: Fishbone diagram (Silvers, J. 2004.)

When the session to collect the actual root-causes is organized soon after the quality issue is solved, it can provide more accurate reasons and more concrete requests of what to do, than other types of session collecting feedback.

Imaginary example for testing questions:

- *Did you test it? - No.*
- *Why? - I didn’t know feature was there.*
- *Why? - Ummm… difficult question. I don’t know.*
- *Did you not see it in planning meeting agenda? Did you not see it in sprint backlog?*

- Okay, I did not join the meeting, testers are not invited. I read the backlog, but it is not planned for this week but next one.
Dialog would then continue asking why, to also see why the backlog is not up-to-date and why testers are not included. Outcome to record and share from this could be changing the meeting practice to include at least one test team representative, and improve other types of collaboration between developer and tester.

Imaginary example for development questions:
- This error, is it possible to find it with static analysis tool? - Yes.
- Did you run the tool? - No.
- Why? - It is too slow. I fixed too more errors the same day, and it would not have been possible to do that if using the slow tool.
- Why the tool is slow? Anything that can be done to make it faster? - No. Maybe not. We don't know …
- Okay, I'll take it on my task list to ask further from others. Then was there any other way to find the error?
- Yes, anyone can find it when using in real environment. Reviewer also found it, but did not report it.
- Why???
Dialog would then continue asking why the reviewer and tester failed to notice or report to developer in time to prevent it. Outcome from this is the open question of tool being too slow, that possibly can be changed in future.

With a new feature development two separate teams in different locations were involved. The usual happened, when software was released from both teams, it did not work at all. On the surface both of the teams did right things, both had skilled developers and testers to do their part, there was a regular meeting to do the planning and co-operation. But still this new feature in software failed, error was leaked. It was seen as a good opportunity to make an example how the root-cause analysis would work. It would also be a perfect example, because the teams are not only located in different place, but in opposite time-zones, that is reducing the live communications and focus on emailing, so if there would be any improvements found, it could be applicable for other team co-operation as well.

From past experience the technique of asking “why” was known to work in face-to-face situation. Quality manager would call in meeting with all the key stakeholders who were involved in the process. And to avoid the sense of blaming a person, and the sense of torture when held responsible trying to answer why I and my team failed, the meeting was arranged usually in cosy lobby or coffee room. The stakeholders would be a lead developer, developer whose change introduced the leaked error, developer who reviewed and tested the change, and test team leader. The group was intentionally focusing in the development activities to find out the real causes and real changes that could help building-in the quality. The dialog would go through each person and ask questions deep enough to have a satisfactory answer: there is nothing we could have done differently, or identify one or several things the team can change in the practice or the quality manager can take and escalate in the organisation. This root-cause analysis meeting would be arranged right after the leaked error was fixed, so that all participants and especially the lead developer has deep understanding of the issue happened.

In this new environment, it was not possible to have the stakeholders in same room, just for the sake of one error leaked, one new feature that did not work. So there were two options: To wait until the next regular online meeting and start the discussion there, but the worry was it is too late considering people still have the issue in fresh memory, and the meeting would have more important things on agenda. Second option would be using email, but the worry is that it would not have a true dialog between persons, and with many people in two teams, the emails would be long and nobody having time and energy to dig in the deep details.

It was decided to start by email, so that the question has been presented to save some time, and possibility to continue as discussion in the regular meeting next time if seen important. But writing the email was difficult, for many reasons. The technique had not been used before with these people and this way, so the purpose had to be explained but very shortly. Emphasizing it is about the things gone wrong and the question how to improve. Since the dialog would be slow, waiting reply always to next day, the questions about communication, development and release testing were all included. Also in live dialog it would be possible to start with easy question, and gradually go deeper, but to fasten the chain and to push it some to right direction, also some leading questions were included.

Some people replied the email, but not answering the questions. Some replied to the leading questions, saying good or bad idea. And some agreed those good ideas should be taken in use. But likely the purpose was not understood. Also as anticipated, the next regular meeting did not have this issue on agenda, and the matter was pretty much forgotten.

One of the replies or comments was that between the two teams there is quite often something forgotten in the chain of communication and development. And that it would be good to have a checklist for feature development to review next time, early enough. The QA team had worked on a template/checklist for project and feature development (when investigating test plan improvement needs), but as it was not published, it was not available in situation needed. However, it was a quick task to find the original document, review and slightly modify it against the found issue, and provide it with confidence it can help next time.

Observations of the one occurrence provides some hints how to develop this practice further. While the original face-to-face meeting recorded only the outcome, the email approach tends to record the middle of the conversation, which would be nice to forget, as it is difficult to admit there was a fail and maybe some excuses that would not be nice to expose and share with wide distribution amongst colleagues. It is somewhat easy in live conversation to say “I don’t know”, so the other person can re-phrase or form another question to continue the dialogue. And recording the details of the technique is really not needed, just forming the path from the incident to the root-cause. A better approach would have been to start the discussion as live meeting over the phone/video connection separately, despite the time challenges, to really explain what is expected. After the technique is understood and shared, both teams could do it independently and get together with the results and rationale behind it.

There was also one, perhaps critical fail in the email itself. As usually cross team communications need to have some amount of politeness and diplomacy, and spirit of constructive comments also when need to ask or give negative feedback, to collaborate effectively. This time the email contained a typing error that may have turned the message to finger pointing of people instead of trying to develop practices. “I’m not blaming anyone” appeared the opposite, when word “not” was accidentally left out.

5.3 Improving the Project Execution Phase

5.3.1 First Project Retrospective to Collect Improvement Needs from the Team

The first change introduced in this thesis was the project test plan. The outcome of the investigation was a list of quality issues and changes that small part of the development team had collected. But there was no certainty, if the entire sees the situation in same way. In this situation feedback from the team was needed for actions of the starting project, but there was no practice how to get it. Shortly after that test plan activity, an old project came to its final end; decision to close all maintenance activities was announced in the organisation. Project management theories suggest collecting feedback and lessons learned in the end. There were multiple reasons to get the team together to discuss, and the agile approach of arranging project retrospective was selected for the purpose.

Everyone in the development team was invited, three scrum teams, QA team and both managers. There were several people who were recently employed, and really did not have experience of the oldest project closing, but some experience of the on-going projects. Also ideas for the starting project were needed, so getting all types of feedback was target, everyone's opinion should count. It wasn't only the feedback what was needed, but also suggestions what should be done. So the retrospective session should enable a trusted environment with freedom of opinion, and room for brainstorming and ideas. A scrum master guidebook suggest first to have a summary or sighting to the past project highlights, then time for individuals to think and collect their feedback, and finally discuss in groups.

A big meeting room was booked, to fit everyone. The schedule was prepared for 1,5 to 2 hours. Organizer had prepared a schedule, but the phases were not switched by the clock but by seeing the activities on-going and getting done.

A very short summary of past projects was given by a person joining the team long time; it was not agreed beforehand and only works as opening of the session and time to realize team had really achieved something, to complete the project.

A scrum master guidebook tells the feedback should be collected in simple way, the positive and the negative. But it was a long time in the team since any feedback had been collected and big changes happening with more projects and more employees. The expectation was it is difficult to get started, if there are no real questions to provide comments about. As QA team had previously discovered, the needed changes can be roughly categorized in three levels: project, feature development, and daily work. These three categories were given as a guideline to start with, as described in table 2. The positive and negative were also phrased in several ways to allow all types of comments: Good / positive / things to keep; Bad / negative / things to change.

Each individual was given pen and paper to write down feedback with the help of the categories. The task was to be done alone, without discussing. To avoid distractions of email, chat and thinking of the work tasks in hands, nobody was allowed to take their laptop in the session. Meeting has a clear focus on reflecting the past and thinking freely, so it is important to minimize the distractions.

Often the feedback sessions get a lot of comments about things that are not running well, but what the team itself cannot fix. Like limitations in tools, some other team's practices, that can be maybe requested by escalating the issue within organisation, but no direct way to change it quickly. This type of information is valuable feedback. The first, individual part was meant to collect any kind of feedback to know what are the items bugging people (and might need to be escalated).

Second part was to form small groups to share and discuss their written comments, and as a group to make 1 to 3 suggestions what to improve and how in practice. The group should also prioritize the suggestions in the categories, like presented in table 3. This part had a purpose of really looking forward in what team itself can do to improve, to take initiative and commit to change plan: "we can do better and we want to".

These groups were given a time, during which they were allowed to do the discussion anywhere they wanted: some group would stay in the meeting room, some go in coffee room, and one group even found their way in the empty hobby room nearby. Organiser of the session visited the groups, to listen to the progress, and finally notify the time is running out, for everyone to come back.

TABLE 2: Form of collecting the individual feedback in three categories

Project	*Feature*	*Work*
good	good	good
bad	bad	bad

TABLE 3: Form of collecting the group actions, three prioritized changes to do in the three categories

Project	*Feature*	*Work*
1. to-do	1. to-do	1. to-do
2. to-do	2. to-do	2. to-do
3. to-do	3. to-do	3. to-do

Entire team back in the same premises, each team was allowed to tell the most important thing on their list to change. The intention was to have more items listed, discussed and prioritized, but the plan was changed due to time running short. It was already notice when first group said their first item, starting a lively discussion and many ideas. So each team was only given chance to present one thing. Items were listed with a responsible person to drive it forward and a target date for check-up.

From QA team perspective, the items were supporting the assumptions that existed in their meeting minutes and plans from earlier, and also completed some gaps. There was new material and task list to work with. All written papers were collected, so that both individual feedback and group work is saved. Agenda of the retrospective, action list, and summary of the feedback was shared as one document in shared drive, so any team member can go and re-collect the ideas. And the format can be reused.

From manager’s request there was also one element not foreseen: request to present the positive feedback in monthly team meeting. As important as it is getting a task list for doing improvements, everyone should be reminded the things to keep and things to make the workplace a good place to be.

Observations show that this known but forgotten agile practice is needed in the team. Discussion always needs to have enough time, for each person to say their opinion, and for the topics to evolve and form into real actions. This method was used by the big team first time, and the format was constructed by organiser alone slightly different from guidebooks or any discussion, in order to keep it a secret - an element of surprise to enable the brainstorming spirit. Even though anticipated in the agenda, the time run out, but it did also reveal that the team has a lot to say, and to discuss together. In current working model there is no other situation where entire team could address these things. Team seemed to have a trusted environment already, so the plan worked well. And the addition of reviewing summary of positive feedback was also welcome, interesting to team and managers.

5.3.2 Second Project Retrospective to Evaluate Improvements and Continue Collecting Improvement Needs

Three months after the first retrospective, the time planned for this thesis research had passed. More importantly, projects had been progressing, and it was seen a good time to have retrospective again. All members of the team had experience of the implementation start phase of a project. Also many changes were done in that time, so the feedback and review was seen important, in order to set the course again. Many changes involved those activities mentioned in this thesis work, but also all scrum teams and the co-operative teams' activities well.

The idea again is to see if the exact same issues rise again with same volume, so they should have more priority in future. It is expected there are new comments as well, on things done differently or anything that just was not remembered or mentioned. The scope is also different, since now the time period to evaluate is same three months for everyone, the first retrospective not having limitations and difference of newly employed months to those having few years in the company. Results are not meant to be directly comparable, but doing the comparison can indicate if there were changes and if they were successful or not.

Again, all team was invited to same meeting room. The element of informality and a little surprise was in the beginning of the session: small celebration with the opening words highlighting the success of latest project milestone achievements. Opening words was the starter, just like previous time.

Looking back also now included review of the previous retrospective results, so the summary list was shown with actions agreed and nominated responsible name. Task list is shown in table 4, but the topics are generalized for confidentiality reasons. Some of the actions were clearly about addressing an issue that was only urgent at that time, and is not going to be continuous issue. Few issues got also actions done. But most issues got no actions done at all, or partially started but still continue to be an issue. So already before starting the part of collecting new feedback, it was realized that most people forgot the plan as it was not followed during the months. Also it was not enough that one person is nominated as responsible to facilitate the issue, but the team needs to commit together to make the change happen or reschedule/de-prioritize the change together. Item was added to task list of this session, to have the retrospective actions list on agenda of team's regular meeting.

TABLE 4: Task list of first retrospective with remark if the action was handled by the time of review in second retrospective. (Not exact items, but generalized to be public.)

R1-A1	Regular issue of team meeting practices, certain information is not shared frequently enough. Agreed action solved once as agreed, but nobody followed the practice to become regular.	Solved
R1-A2	New feature development issue.	Not solved. No new features done anyway so this item did not become important during this time period.
R1-A3	Definition of done in task level unclear.	No meeting to discuss arranged. Development teams did not raise the issue during this time period, suggesting the daily work does not suffer from missing it or found a practical agreement.
R1-A4	Feature level definition of done to contain better information sharing to customer.	No meeting to discuss arranged and this item does not have clear owner, as the team is not directly involved with customer.
R1-A5	Tool issue for few people.	Solved, required only simple actions for those individuals.
R1-A6	Issue with a tool, task to collect data for escalating the issue.	No data much collected, but issue was occurring at the time and not happened again.

Since the time-period to review was shorter and experiences of previous session the agenda was changed a bit to be more simple and quick. Instead of asking different types of feedback in different categories guideline was only positive, negative and neutral or something between. The expression was again also formed in different ways, the feelings were added from a scrum master guide book: positive = I'm glad, between = I'm a little sad; negative = I'm mad. The neutral value was added, since many of comments in first session were formed as "this is good, but…".

The individual part was conducted same way as before, everyone to write their notes with pens on paper in silence. The group conversation then continued differently, all in

the same room: first by reading each other's papers and adding comments, and after that to start discussion. The group was again asked to collect a list of concrete actions that team should change in near future. This time all groups were asked to present their proposals quickly, after which all team discussed, and the task list was collected. This time there were more items on the list, a bit more concrete. Some items were new compared to the task list from previous retrospective as described in table 5, although those could have been in the feedbacks before. The session was closed in about the 2 hours as planned.

TABLE 5: Task list of second retrospective, with remark if item was new or similar with first retrospective. (Not exact items, but generalized to be public.)

R2-A1	HW in investment request to escalate. Action to manager to continue.	NEW
R2-A2	Common request to have some training and best known methods available and shared in wiki. No responsible person for any detail topic.	NEW
R2-A3	Improvement wish for test automation, but no detail requirement nor specified owner.	NEW
R2-A4	Daily scrum meeting notice to one team, to make clear separation of the quick meeting and discussion to continue in other time.	NEW
R2-A5	One key person found as bottleneck, since the actions done in that role is not clear to substitutes. Real proposal for substitutes to have more hands-on experience regularly at all times.	NEW
R2-A6	Key person responsibility to be more clear for teams to handle their part in more organized way, one counterpart of the person to start building more efficient collaboration. Action started in few days of the retrospective.	Second proposal for issue mentioned in R2-A5.
R2-A7	CR process definition still needs actions. Owner and few key roles nominated to handle since they should have been the owners already (but in practice weren't). One practical action mentioned and done by collecting the materials from before on the same day retrospective was held.	Partially same as R1-A2.
R2-A8	Practical request to change regular meeting to be more frequent.	Same as R1-A1 but in other words.
R2-A9	Issues with error handling. Real examples causing trouble to developers. Error manager nominated as owner of the item. Part of the request is a more detailed wiki, which was released in few days after retrospective.	NEW
R2-A10	Retrospective items to be followed in regular team meetings.	New item, found during review of the first retrospective results.

Observations show that even if the practice was found useful the first time, and there was a form to run the session, the continuous improvement does not happen by itself, and the practice needs to be evaluated together with other practices.

Session is not long enough for really digging into details, what changes and how to do it. Some discussions afterwards reveal there should be more time to discuss thoroughly, and analyse. On the other hand, this kind of discussion needs could be added in the task list as open assignment. The scrum master guidebook instructs for all team to vote for the priorities in the task list. Both sessions did not have time to do it, but this does not seem like a problem since the topics are in very wide scope and does not involve all team to do changes in their practices. Many actions are also left in general level, so that it is hard to nominate a responsible person, but if the list is reviewed more frequently in team meetings, it should be enough to keep planning the changes needed. The amount of feedback was big in both times, so the list is still "a tip of an iceberg". The wideness of topic and amount of comments tells that maybe there is no other forum for team to give this input in a way that seems to have an impact; also there is an interest for all to develop the practices, and that having this kind of retrospective meetings should be a regular practice. Three months was a good period because already then many things had changed, and the old results were forgotten.

There are also other places and ways for team members to provide feedback. Like the scrum teams can have their own retrospective every sprint, practically every two weeks. But this session of entire teams enable different perspective not only because of the time and focus difference, but about working together. For continuous improvement the sprint retrospectives still have a place aside the project retrospective, and there could be place to evaluate how the improvement ideas and progress are to be handled between these.

The results showed that the most important issues affect all scrum teams almost equally, so the change should benefit all. Some comments describe this situation, like "I didn't know your scrum team had this issue, we also have it so we should really prioritize it." Also as scrum team can have different practices the discussion can be really effective collaboration method, starting with comments like "I didn't know your scrum team had this issue, this is how we solved it and I think I can help you with that."

Observations after the retrospective also revealed some challenges. The team has a good trust level and open communications, but there can be situations of almost crossing the line. The purpose is to find out how team can do better, but when something has failed that seem to rely on one person's role only, there is a risk of discussion going on too personal level in the comments. Brainstorming and open place for feedback cannot mean sensitivity and respect for the individual is forgotten. Follow-up of the session is not only important for the tasks, but for other possible impacts as well.

The entire team retrospective seems to be useful practice to collect feedback and empower the team to come up with solutions for continuous improvement. The finishing work also takes time, collecting the hand written notes and typing them in a document to save. While doing this, it requires reading every note, and enables the possibility to form some overview and maybe analysis, too.

6 NEW IMPROVED PRACTICES

The objective of this study was to find ways to improve quality by improving working practices in software development team. Some practices impact directly to the product quality. Other activities help software projects to succeed better, building in the quality already inside the workflow.

Project retrospective was found useful practice to keep. It provides positive and negative feedback, and list of changes and actions needed. When carried out frequently, it enables continuous improvement with new ideas and evaluation if previous changes have been successful or not. Making the positive feedback summary visible was found important, in order to know what things are done right, as the improvements are focusing only on the negative comments. Since the entire development team participates this session together, working together towards shared goal also builds collaboration and team spirit. As the team, including the manager, is making the decisions, unwanted changes that are not seen important are avoided, building commitment and motivation to also improve practices.

Customer perspective was introduced in a form of **exploratory testing**. Highlighting this purpose on the first occurrence made it easy for everyone to quickly understand the benefit of this method: finding severe errors quickly and having an overview of the unexpectedly poor maturity would have been a shameful moment if customer would have been in the premises. Exploratory testing impacts to the end result quality directly and to prioritization of remaining tasks in the project. It is also a way to give suggestions and ideas about the features, and perhaps improving current of future software. This practice is found useful and will be used, perhaps not regularly but when targets are seen important in any project implementation phase and developer testing coverage needs to be complemented.

Checklists for project and feature **planning** were collected, to remember all relevant activities needed to cover or review. New practice would have been to start using them, and perhaps review the project progress status also against the items. But in the timeframe of this study there was no real usage that could have been showing any re-

sults. There were difficulties in collecting the information. But all the preparations and progress could be seen as first iteration for the practice, and there is a strong wish from development team to have this kind of lightweight document support for the planning and development process.

Project status sharing regularly to development team was well received. There were no complaints of excessive use of email for this purpose. Some feedback was given to share even more details, which was done occasionally to complement the message. It was not really monitored how the information was used, but the feedback especially in the implementation finalization phase having the key figures and dates easily available helped in sprint planning and daily task prioritization.

Continuous improvement can be included in development team activities by sprint and project retrospectives. However, the results focus on opinions what is important for employees and practices used, and what could be important for improving quality. To investigate the actual problems more deeply **fishbone method** was introduced to find detailed root-causes and possible solutions. The method was used only once, not inside the team but in situation of two teams contributing to create a new feature together. Some solution proposals were found, but due to the setup it was not concluded properly. Using the fishbone method for root-cause analysis could work better and will be introduced again with better preparations and structure.

The amount of preparations when introducing a practice first time also played important role in matter of success. Exploratory testing was well prepared and organizing was done based on strong experience that paid off as success of introducing the practice. Also the retrospective relied on knowing how the session setup could be done, but part of the preparations was in fact done when first conducting interviews about project planning and making conclusions from the collected information. Project status sharing was simple task as the status report had been created before for management purposes, so only slight modifications were needed. Using the fishbone method was found difficult, because the setup was not well planned: the method could have been more useful if it was used in face-to-face situation, but there were no extra preparations done for the challenge in handling it offline by email.

Finding the sources of improvements was found essential factor working with the improvement goal. When the information of problem is given from outside, in this case a manager request for a test plan in the very beginning, the improvement investigation needs to start from verifying if the problem exists. It will take time to proceed further, to methods and practices how to improve. Monitoring (the process) was used as source for using the Fishbone method, as a quality violation occurred in the software release flow, causing severe error in the feature. Retrospectives are source for improvements that do not need explanations or reasoning to the team, since the feedback and improvement requirements are created from inside. Background theories also form a source, like the project status sharing, that was not originally requested in any way, but turned out to be needed.

Improvement approach was iterative, plan first then try it in action, and continue cycle after evaluation and small changes. Practices were not introduced all at once, but one-by-one when appropriate target was detected. Exploratory testing was planned to be introduced, as from experience it can be used for many different purposes. The plan was on hold until the situation where it provided answers for the problem where development team did have different perspective to software maturity than the status report based on testing activities showed.

7 DISCUSSION

7.1 Conducting the study

The study was conducted with a mission of finding methods and practices that can help the team to build in quality. The focus was in the questions of what practice, and how to organize it, how it is applicable in the environment. Theoretical background that is part of the thesis was not included in study phase in a way that the whole team should increase the knowledge. After all, all members of the team already have understanding and working experience in the industry. Instead the practices were introduced with "learning by doing" approach, with expectation that once the new practice is tried once, it will produce results that speak for themselves and there is no need to justify the action from any theory. But that approach also included finding answer to another question, when. Even as some practices were known to be useful, they were introduced only when finding a proper time that fits in schedules, and enables providing results.

Making permanent improvements takes a lot of time and effort. Introducing a change can be successful and appreciated, but doing it once or even twice does not tell if the change is to be kept and what are the impacts. Both success and fail are valuable results, as long as both are used as learning experience: to analyse and identify reasons in this environment, to be able to develop further.

None of the introduced practices failed completely, considering using it had an impact and found useful to address some quality issue. Using the project retrospective can be also a practice that includes the evaluation and planning part for any changes, without adding extensive bureaucracy for process improvement. It is also planned to be a permanent and continuous practice in the team. Therefore it can be seen as most important finding for continuous quality improvement. The reflecting, analysing and learning is the important part described in process and quality improvement theories, and this is the one practice that now really uses that approach.

Pain points are still the contents and using of checklists for project planning, and using of the fishbone method. Both require more work and more iterations, and possibly even

different approach before it can be said they actually produce quality improvements. Despite other challenges, the fishbone method could have been better digested if the method theory was shared in advance.

None of the changes were relying only theoretic without real experience. Some references were collected from other software companies, and for example agile practices were studied together by some members of the team. Study did not include really benchmarking, and not even comparison to other teams in the company. Since project planning is something every team does in the company, interviewing any other team could have impacted the outcome positively.

Involving the entire team also had a good impact on result. Including all team does not necessarily mean that everyone needs to sit down in same room to discuss and decide. Heiramo (2013) describes non-hierarchical working practice to work well in agile environment: one or more persons need to lead the activity, but instead of asking permission (from manager or superior or colleague) one should take responsibility of representing their ideas, putting them under review for others, collect the feedback and decide what feedback is essential and need to be taken into account.

The members of the development team did not know the observations were made for study and thesis. The target of the work - to create better practices - already existed anyway. From this perspective it seems plenty of changes and improvements can be done when working together towards common goal, as people are motivated and empowered.

7.2 Success Factors for Introducing Improvements

As product (software release) life cycle is handled as project, and making of new software is a continuous process, the theories of project and process management do provide relevant material to base the planning. Viewing the literature of the common mistakes can easily provide quite direct answers. Like not having a clearly defined target is a common failure that was also noticed as a challenge for creating project test plan. But the theories and basic descriptions were not enough to apply the practices. The success of used methods and techniques involved experience, a living example how to use them.

Therefore it is expected this thesis work does not tell anyone else how to do the same changes, but the descriptions can provide a slightly better insight, to work as an example.

Experience of used methods had a clear impact of success rate. The writer of the thesis had been organizing exploratory testing sessions for several years in other type of environment: It was a regular test method for testing team to fill known gaps in test coverage. The method itself is flexible and can server for multiple purposes, so it was relatively easy to apply in the new environment once the goal - making the development team aware of end user quality - was identified. Also the project retrospective format was known from participant perspective, in multi team environment where the practice was running for several years.

Already beforehand both of exploratory testing and project retrospective were studied and analysed in small scale, to have an understanding of the key elements that made them successful practice. Those methods, however, were not taken in use directly, but with careful planning to fit the new environment. For example retrospective according to guidebooks and experience should take 3-4 hours, but in this case when people seemed to complain of any additional meeting is time away from real work, it was done in less than 2 hours still including the discussion and brainstorming element. The root-cause analysis dialog was not successful on the first trial; there was some experience in participating that type of session, but the environment and arrangement were so much different, that it will require many changes to be better applicable.

Agile approach of doing together seems to be helpful, as the ideas evolve to direction that may have been not foreseen. People have different experiences and different knowledge that always impacts the perception. Doing together also impacts positively the team spirit, attitude and desire to do together and improve. While emphasizing the team itself is doing good work, there is a risk of starting to feel superior compared to other teams. Therefore sharing and collaboration also needs to be built, in order to achieve the final goals of making the business successful.

Finding practices that require very little effort, and therefore save costs in working hours are also well adopted by team. Like there was prejudgment and minor resistance

against project retrospective on the first time, since it is taking few hours from everybody. But not anymore the second time, as it was proven not to be wasted effort.
Another example of improving practises with no impact on scheduling would be one scrum team organizing their regular weekly discussion with equivalent team in other location as video conference, instead of previously used audio call only. This example did not require extra effort of organising it, since all other things in the meeting arrangement were kept. But this practice is not included in above chapter to explain changes, since writer was not there to do any observations. The impact and improvement was highlighted in retrospective comments, making it visible for others not taking part, and as a good example for others to follow.

7.3 Limitations and Restrictions of the Study

The organisation and team being new, there are some limitations for doing improvements. Processes exist, but those are not defined or shared in level that supports understanding the current practices. Process descriptions could be also used to see/analyse if there are any obvious gaps in it, or differences between definition and the actual behaviour. Process and change management guides to understand the current situation, describe the target, and define actions from the difference. Both current and target situation were not used in this study, although this principle was acknowledged. Practically the chronologically first improvement included variety of discussions just to understand the current situation and known problems in it.

The lack of process descriptions is not raised as quality problem in the scope of this thesis. There are no references to actions planned or on-going in that area either, as not being relevant in this scope. It was decided to focus on actions inside the team, and also have results in defined time period, so involving other teams that process definition work requires, is not part of the study.

Decided time period can also include a variety of other practice and quality improvements, inside the team. Only practices and methods that are found significant, have a purpose and are actually monitored closely by writer are included. This thesis content does not therefore provide any insight about how much changes and new practices can

be introduced for a number of people. For example developing daily sprint retrospective practices are excluded. Also actions containing specific details of the particular software are excluded for confidentiality reasons, and for not producing general information.

Any innovation guide tells also some reality, there can be hundreds ideas out of which perhaps just one will success. Recording all those ideas is important for evaluation. All the ideas and changes are not included in this study, as that information would be probably beneficial for the team only. Also this kind of list is not produced, but the proposals are remembered in team members' minds. It is not possible to do many changes all at once, so having this type of list could be needed in future for bringing the ideas visible.

Monitoring and analysis in this thesis are not done as team work together, so it is possible that writer's opinion has impact on results. This thesis work content is reviewed by two members of the team, to avoid any obvious conflicts.

7.4 Evaluation of the Thesis Process

To begin with it was not obvious what is the working title, the focus and limitations of this this study, which can have impact on the consistency of this work. Quality and testing improvements were the first objective. During the interviews and discussions when trying to establish a picture of current conditions, it was noticed that testing is not the one big quality issue that needs the kind of changes that makes a big impact. Instead there seemed to be many smaller things that required small improvements in everywhere in the project lifecycle, and already many ideas of practices that could be introduced.

Part of the challenge was that it was not only the author of this thesis, but other team members as well that were recently employed, and it is time consuming to learn all the practices and contacts of the new environment - and perform the regular tasks as well. Doing the interviews and discussions were not always targeted to solve particular issues for quality, or doing other tasks, but all they worked for the benefit of being part of the team and finding a place in the current environment.

Some limitations were needed to have material for just one thesis work, and to produce it in desired time. Working directly inside the development team allowed direct communications with team members and doing the observations easily, so that was an easy decision. Choosing only agile methods could have been one option, or expanding it to any participatory methods that also build team spirit and cooperation channels between individuals. Since the original topic was about quality and testing, but quality management and other theories emphasize the importance of quality in all phases and the earlier phases improvements can have good impact on costs, it was decided to turn the idea "upside down": make it visible what kind of changes can be done in development team itself.

The title contains "software development" and "evolving" as they are expected to be somehow limiting factors or essential background information about this study. Evolving or forming status of the team is a fact, but not necessarily valid information considering the found practices could be just as well used in more stabilized environment with improvement needs. The practices could be also applicable in other types of companies. This title is still kept, since it probably still explains the existing working model well.

7.5 Conclusion

In conclusion there were several improvements introduced to the working practices of the team. Most of them are considered to be permanent and usable in future, and all of them can be still developed further. This thesis describes the practices, with background theories and information of the case, providing an example for readers to see if the practices could be applicable for other teams. In addition, evaluation of how the changes were introduced can be seen as valuable example to anyone starting to do similar development. Practices and the evaluation was not only one time effort, but enables continuing quality improvements in the team. Objectives of this thesis are met.

8 FUTURE DEVELOPMENT PERSPECTIVES

This study - the work done in the company and reviewing the theories - has given a lot of material to continue further. Plan is to use the information to discuss and decide overall quality targets, focusing on the product quality and customer perspective more. Collecting a backlog for the improvements ideas already available is also a next target in the work environment, but the known team specific topics are not essential to be included in this study and not listed here.

It is not decided yet if there should be some tracking the achieved improvements in some consistent way. Of course from retrospectives and other forms of feedback it can be detected there is progress. One indicator about success is project success, if projects and their quality are comparable. As one of the findings was that positive feedback and what we are already doing right is equally valuable information as improvement ideas, there could be some benefits in monitoring and collecting successes in practice level also.

There is still work to do with the currently found practices, to develop further. In addition there are many process improvement methods and tools that can be someday found useful. Studying also other agile ways that exist already or are forming is probably the way forward, since the principles and participatory methods seem to have positive reactions.

Looking at some process and quality improvement materials, the topics escalate to guides about overall organisational development. Building the team spirit and collaboration are few examples. In newly established company things like learning and getting the tacit knowledge into use are perhaps not the most important things to take care of. But already now with practice improvement phase these topics have raised as possible key factors ensuring the quality in future.

This study focused on one team perspective, but already there are some references to organisation level topics. Collaboration in the company is essential, and some process

and practices improvements are already on-going. One objective for this thesis was also to present the findings in way that other teams inside the company can see as an example. The publishing of “best known methods” needs to be planned. Working models like workshops and meeting face to face with representatives from different teams are good ways to co-operate. It could be also enough to have one visitor from another team to take part of some practice, or one of this team to facilitate a used practice in another team. All development team improvements need to work in the environment it exists, sharing information is a must, and with synergy also effort can be saved. Company direction in tools, practices and quality improvement will guide the way.

REFERENCES

Ambler, S. 2009. Agile Testing and Quality Strategies: Discipline Over Rhetoric. Luettu 15.9.2013.
http://www.ambysoft.com/essays/agileTesting.html

ASQ 2013. Quality glossary. Luettu 18.9.2013.
http://asq.org/glossary/

Black, R. 2007. Pragmatic Software Testing. Becoming an Effective and Efficient Test Professional. Wiley Publishing.

Cobo, J. Ortiz, I & Mataix, C. 2010. Design of a competence-based model for managing programmes and projects. Project perspectives 2010, 15-19.

Eskola, J., Suoranta, J. 1998. Johdatus laadulliseen tutkimukseen. Tampere: Osuuskunta Vastapaino.

Heiramo, P. 2013. Futurice blog: Seeking permission, or feedback? Luettu 15.9.2013.
http://blog.futurice.com/seeking-permission-or-feedback

ISTQB. 2013. Glossary of Testing Terms Version:2.2. Luettu 17.7.2013.
http://www.istqb.org/downloads/viewcategory/20.html

Laamanen, K. 2009. Johda liiketoimintaa prosessien verkkona: ideasta käytäntöön. Helsinki: Laatukeskus.

Lehtimäki, T. 2006. Ohjelmistoprojektit käytännössä. Helsinki: Readme.fi.

Leppälä, K. 2011. Projektitoiminnan musta kirja; miten aikamme menestynein käytäntö saadaan takaisin raiteilleen. Helsinki: Readme.fi.

Myllymäki, R., Hinkka, T., Dahlberg, T. & Uimonen, B. 2010. Miksi tietojärjestelmäprojekti epäonnistuu? Tositarinoita tuhon teiltä ja onnistumisen siemeniä. Helsinki: Laserpaja Oy.

MSI 2011. Juran's trilogy. Luettu 18.9.2013.
http://msi6.com/MSI6/QualityZone/QzoneJuranTrilogy.aspx

Pelin, R. 2009. Projektihallinnan käsikirja. Helsinki: Projektijohtaminen Oy Risto Pelin.

Phillips, J. 2005. IT-Projektinhallinta -sertifikaatti. Helsinki: Edita Prima Oy.

Silvers, J. 2004. Cause/Effect Analysis. Luettu 15.9.2013.
http://www.juliasilvers.com/embok/Risk/RiskAssessmentMgmt/causeeffect_analysis.htm

SkyMark Corporation. 2013. Luettu 26.9.2013.
http://www.skymark.com/resources/leaders/ishikawa.asp

University of South Australia. 2009. Project Management Methodology - Project Lifecycle and Phases. Luettu 15.9.2013. http://w3.unisa.edu.au/ists/governanceinit/projectmanagement/methodology/projectlifecycle.asp