CHALMERS
UNIVERSITY OF TECHNOLOGY

Why do developers struggle with documentation while excelling at programming

An empirical evaluation of the motivation for software documentation

Master of Science Thesis
in the Management and Economics of Innovation Programme

DANIEL FONTANET LOSQUIÑO
TOMAS URDELL

Department of Technology Management and Economics
Division of Innovation Engineering and Management
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015
Report No. E 2015:042

Master's thesis 2015:042

Why do developers struggle with documentation while excelling at programming

An empirical evaluation of the motivation for software documentation

DANIEL FONTANET LOSQUIÑO
TOMAS URDELL

Tutor, Chalmers: Jan Wickenberg

Department of Technology Management and Economics
Division of Innovation Engineering and Management
Chalmers University of Technology
Gothenburg, Sweden 2015

Why do developers struggle with documentation while excelling at programming
An empirical evaluation of the motivation for software documentation
DANIEL FONTANET LOSQUIÑO,
TOMAS URDELL

Department of Technology Management and Economics
Division of Innovation Engineering and Management
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 (0)31 772 1000

Chalmers Reproservice
Gothenburg, Sweden 2015

Abstract

In software engineering, technical documentation is one of the activities that play an important role for the success of software projects. When developing software systems a lot of time is spent not only implementing your own solutions into the code but also understanding and maintaining the code solution produced by others. However with an emerging agile community the priority and engagement in documentation may stagnate and effort invested into those activities may to some extent be considered inherently wasteful. Therefore, the current perception for many software engineers may be that technical documentation is incomplete and not updated.

An important aspect of the issue of producing documentation is the motivation of the developers who are responsible for its completion. A question that might be asked is if documentation overall is an activity that the software community would be better off without, or if it is a neglected activity with room for improvement.

For that reason this exploratory multiple case study aims to identify some of the issues to shed some light on why technical documentation has got into such an unfavourable position in the software industry. The study investigate the motivation affecting software engineers when producing software documentation, how this motivation evolves over time and why there is a difference compared to other activities in the software industry.

The study consider prospect theory as an explanatory theory for how improvements in outcomes are perceived in terms of value. In addition this study also analyses how experiences are remembered and evaluated, as it relates to how excitement is considered for similar tasks in the future. The study can affirm that software engineers' motivation seems to be affected by excitement and value dimensions when it comes to producing technical documentation. Additionally, in the long term, receiving constructive feedback can improve the excitement and the perceived value of writing technical documentation.

The study finally observes that developers to a higher extent seem intrinsically motivated by the coding activity than the documentation activity, which the study points out as a primary enabler for the difference in behaviour between the coding and documentation activity.

Keywords: Agile Environment, Experiencing self, Feedback, Self Determination Theory, Prospect theory, Remembering self, Software Documentation, Software Engineering, Technical Documentation.

Contents

List of Figures

List of Tables

1
Introduction

When developing a software system a lot of time is spent not only implementing your own solutions into the code but also understanding and maintaining the code solution produced by others. To understand the code and its properties has for a long time been at the core of the programming profession and is vital to the progression of a successful software product (Brooks, 1983). Comprehension of code is something that has historically gained the attention of the software engineering community, perhaps due to its importance for a wide range of software engineering activities (Brooks, 1983).

The concerns raised regarding comprehension and maintenance of code are not to be taken lightly upon as the time investment is often considerable in comparison to time spent implementing code. Furthermore failure in understanding the code and maintenance of its properties can often be the cause of failure for entire projects (Sommerville, 2010). The methods ensuring understanding, collaboration and communication is therefore of utter importance to the success of a project. Technical developer documentation is one method to ensure that reliable communication is maintained throughout the project (Forward and Lethbridge, 2002). Documentation has throughout the evolution of software development been an essential component in ensuring quality while introducing new working processes (Sommerville, 2010).

However with an emerging Agile community (Cockburn, 2007), driven by a manifesto stating working software over comprehensive documentation (Agile Alliance, 2015), the priority and engagement in documentation has stagnated and effort invested into the documentation activity is to some extent considered inherently wasteful (Stettina and Heijstek, 2011). The base principle seem to have evolved from working software over comprehensive documentation to a standpoint about comprehensive documentation and working software as two mutually exclusive properties (Stettina and Heijstek, 2011). Furthermore the adoption of this principle seems to transition developers to focus less and less on documentation without anchoring its consequences on future software updates or maintenance (Stettina and Heijstek, 2011).

Although the requirement of producing comprehensive documentation in some project may cause trouble, according to Briand (2003) this is most likely caused by rigid documentation processes with big overhead disregarding the limits of a project (Briand, 2003). Lethbridge et al. (2003) furthermore argues, that there might be a necessity to find a creation process for documentation with greater emphasis on simplicity, conciseness and availability (Lethbridge et al., 2003). However there is little support

that comprehensive documentation in itself would exclude delivering and maintaining working software. Moreover, Lethbridge et al. (2003) points to the fact that most projects that fails or run over budget do so, due to shortcomings in requirements elicitation and poor management. Additionally it is emphasised that outdated documentation provides value if the higher level abstraction remains valid (Lethbridge et al., 2003). Briand (2003) also address the question and identify that also outdated and incomplete documentation is consulted and found useful (Briand, 2003).

An important aspect of the issue of producing documentation is the motivation of the developers who in the end are the ones responsible for its completion. However motivation does not only vary in its amount but also in its orientation, and Ryan and Deci (2000) report of two basic different types of motivation, intrinsic and extrinsic. Everyone prefers to work without external pressure, to carry out tasks and to value and self-regulate those different tasks, to be working intrinsically motivated (Ryan and Deci, 2000). Intrinsic motivation is when a person does a task or an action by choice or self-interest, and while we would like to believe that employees are intrinsically motivated nevertheless most of the tasks people performed in a workplace are extrinsically motivated (Ryan and Deci, 2000).

Extrinsic motivation is instead based on external stimuli or the pursuit of a separable outcome, however extrinsic motivation can be something as simple as recognition for good work. Extrinsic motivation is needed to ensure that workers carry out tasks that are valuable for the firm (Ryan and Deci, 2000). However extrinsic motivation does not have to be bad and employees can indeed be satisfied in a work environment where they are extrinsically motivated. Ryan and Deci (2000) also report of an internalization process of extrinsic motivation where the experience and behaviour of the individual in the final stage becomes very similar to that of intrinsic motivation. However, sometimes is not possibly to mitigate this external pressure and developers will have to perform tasks with forms of less integrated extrinsic motivation (Ryan and Deci, 2000).

A question that might emerge is also how documentation is treated at the university level, where future software engineers receive their final training before entering the professional world. Larsson and Persson (2003) in their studies of documentation at the Swedish universities found that documentation is indeed a central part of IT education. Throughout university education the importance of documentation is often stated and practices for how documentation should be conducted is taught (Larsson and Persson, 2003). While the knowledge and importance of documentation exists and is taught to future IT-developers at the universities, there is a reluctance to perform documentation when reaching the industry (Sommerville, 2010). Although some discrepancies in the required education about documentation between different institutions was found. Larsson and Persson (2003) report about a general consensus around documentation needed for maintenance, knowledge sharing and architecture stability (Larsson and Persson, 2003).

So is documentation overall just an obstacle or impediment that the software commu-

nity would be better off without or is it a neglected area with room for improvement. Stettina and Heijstek (2011) points to the latter and report that Agile practitioners find too little documentation to be available (Stettina and Heijstek, 2011). Briand (2003) however points out that the situation is complicated and observe: "Interestingly enough, most people think some form of documentation is necessary but there is usually little agreement on what is needed" (Briand, 2003, p. 1). Additionally Lethbridge et al. (2003) report that developer might perceive documentation as something that you can create once and use, with little need for updates or maintenance (Lethbridge et al., 2003). However the lack of updates might be also be a consequence of the cost and effort estimated by the value judgement that is performed, and also required by many lean processes (Stettina and Heijstek, 2011). Updating documentation might simply be regarded as a task which does not produce enough value to be justified (Lethbridge et al., 2003).

This study aim to address these issues and shed some light on why documentation has got into such an unfavourable position in the software industry. Written documentation who has generally been considered as a vital and integral part of the development process, has with the introduction of Agile methodology been de prioritized (Briand, 2003). Depending on how one choose to characterize the change in perspective for documentation, it can be seen as a neglected issue in the transition from waterfall to Agile methodology. Thus this study should be of particular interest to software practitioners who feel that the shortcomings in documentation has not been addressed appropriately recently.

1.1 Purpose

The purpose of the study is to investigate the motivation affecting software engineers when producing software documentation, how this motivation evolves over time and why there is a difference compared to other activities in the software industry.

1.2 Research questions

The following research questions have been formulated in order to shed light on the software engineers' perception of software documentation:

1. Does software engineers' motivation affect their behaviour to produce technical documentation?
2. How is software engineers' perception of value and excitement determined, affecting the motivation for producing documentation?
3. Are there actions that can increase or decrease software engineers' motivation to produce documentation?
4. Is there an enabler for the activity of producing code that does not exist for the activity of producing documentation?

1.3 Delimitations

In this Master Thesis the researchers attempt to understand and reveal reasons for why programmers do not vigorously engage in writing documentation. Therefore the researchers has decided on conducting a qualitative study. Qualitative research seeks out the 'how' and 'why' of its research subject without focusing on obtaining quantifiable evidence. Because the study relies exclusively on semi structured interviews for the data collection the researchers primarily gained insight into peoples attitudes, behaviours and motivation, feasible for a qualitative study.

This Master Thesis has focused on technical documentation developer documentation for companies settled in Sweden. All developers in the sample are Swedish with primarily similar background in the software industry and all but one are currently located in Sweden. One has to take into consideration that the people working in the software industry have different cultural backgrounds. Answers and behaviours of people are most likely going to be different depending on the culture to whom they belong, for this reason the researchers consider that this case study can only be generalisable within companies in Sweden or with a Swedish culture.

In this study the researchers do not observe the interviewees in their natural workplace but instead rely on the answers provided in interviews. As Feldman (1984) report group norms can have a strong influence on how people within the group perceive and report about events, however only norms which a have significance to the group are usually formed and enforced (Feldman, 1984). The researchers therefore acknowledge the limitation that there might be a discrepancy between what the interviewee report and how they actually conduct tasks or reason about them.

1.4 Disposition of the report

The first chapter introduces the research subject and provides a background into the software industry regarding technical documentation. The chapter continues with the purpose and the research questions of the study are defined. In the second chapter the theory is introduced and reviews the available theory about software documentation. Moreover the theoretical background essential for the Master Thesis such as Agile development and motivation theory is also reviewed. The second chapter concludes with the theory created by the researchers. Following in chapter three the research design and methods applied is set forth, presenting the data collection and analysis method used. Chapter four presents results obtained from the interviews to facilitate the analysis performed in chapter five. Chapter five present an analysis about the findings in which the findings are put in the theoretical context. Finally, the conclusions of the study are presented and summarized in chapter six.

2

Theory

To support the analysis to answer the research questions the following theory has been incorporated in this section: Importance of technical documentation, software engineers attitudes and opinions about software documentation, agile and traditional methodologies, feedback, prospect theory, two selves and peak-end rule, Job Characteristics Model, Self-Determination Theory, and finally the theory, Excitement & Value Model, created by the authors.

2.1 Importance of technical documentation

What is the importance of documentation in programming? In order to illustrate that two quotes are used. First a Chinese proverb: "Ink is better than the best memory" (Shinde, 2013, p. 118). No matter how clearly you have the facts laid out, given time, the nature of the human mind make it an unreliable medium for storing important information. The value of stored information often is determined by the knowledge it can transfer to a person accessing that information and therefore if one can not understand how to access it efficiently it is of little use even if the information is well formulated and relevant (Wold et al., 2013).

Nevertheless producing technical documentation is a neglected activity in software industry. So the question then become: Why is technical software documentation important? In order to improve the quality of a software product, documentation is required (Berger, 2010). Moreover software documentation plays an important role in terms of maintainability for the system (Jemutai and William, 2013). Although documentation has an important role for maintainability, one also has to consider documentation as an important tool for communication and to transfer knowledge of the system it describes (Forward, 2002).

On the one hand, focusing on proper documentation, Parnas (2011) identified different benefits:

- To facilitate the reuse of old code or designs,
- Easier integration of separately written modules,
- To improve the testing and code inspection
- To improve and correct mistakes made in the past.

On the other hand, a rigid documentation process cause reduced efficiency in every stage of a software product's development due to the increased time investment

(Parnas, 2011).

Furthermore Capri (2006) and Akin-Lagida (2013) consider that good documentation makes information easily accessible and helps new employees or newcomers to a system to learn faster. One also has to take into consideration that people is moving from companies to companies, so documentation can help solve problems when the original developer is no longer available to answer questions. Documentation can also be helpful when developers are looking for big-picture or in-depth information about a software system (Lethbridge et al., 2003).

Sommerville (2010) summarizes the uses of documentation as:

- A communication media between members of the development team but also between developers from other divisions or departments.
- A tool to ease and save time for system's maintenance.
- A help to plan, budget and schedule the software development process, primarily from a manager perspective.
- A guide to tell new employees how to use and administer the system.

2.2 Software Engineers attitudes and opinions about Software Documentation

Selic (2009) signals that software engineers prefer to produce code over documentation. The explanation he provides is that code is something that can be objectively evaluated and modified, whereas for documentation it is quite the opposite. This preference for code over documentation is mostly applicable when the documentation describes and contains detailed information about the implementation. Documents containing detailed information about the implementation also quickly become obsolete, and the effort made to produce and update that documentation is futile (Selic, 2009).

The primary purpose of producing documentation is generally aimed to help your colleagues; to help them understand the code and its interdependencies. Documentation is rarely produced for its authors' sake, although the process of explaining things to others can help the author of documentation to clarify and organize his or her thoughts (Selic, 2009).

Software engineers confirm that they often do not maintain documentation, and when they do that is generally several weeks after changes are made to the code. According to the interviews that Lethbridge et al. (2003) carried out, almost half of the participants agreed and a quarter strongly agreed with the statement: 'Documentation is always outdated relative to the current state of a software system.' (Lethbridge et al., 2003) One of the biggest problems is then to keep documentation in sync with the software; which is tricky due to the fact that software is a highly dynamic entity. Furthermore the opportunity cost of producing documentation should

be taken into consideration as the same time and resources could be used to produce more code. In addition developers also consider code as it's its own documentation, meaning that the code and the code alone contains all information you need (Selic, 2009).

According to the study carried out by Lethbridge et al. (2003), more than half of their participants considered that software documentation can be useful although it might not be up to date. Also, the participants answered that documentation is useful when the level of abstraction is higher. One of the statements that one of the participants said was: 'The documentation is good for giving you a high-level understanding of how the feature is really intended to work.' (Lethbridge et al., 2003) The accuracy of documentation is also highly dependent on how recently the documentation was produced and the amount of changes that has been made to the source code recently, as the discrepancy between the code and the documentation tend to grow with greater source code change (Lethbridge et al., 2003).

Also, the participants of the study admitted that they most likely use and trust documentation that describes a specific feature. Thus the closer you get to the code, the more accurate the documentation must be for software engineers to use it (Lethbridge et al., 2003).

However, according to Selic (2009), documentation that is at the same abstraction level as the code and describes what it does is senseless because you are just trying to duplicate what it is already in the code, and then these comments are useless. Selic (2009) insists that documentation should be produced at a higher level of abstraction and encapsulate the application's concepts and requirements. However design rationale documentation helps software developers maintaining consistency when making precise design decisions. Design rationale documentation also need to include rejected design alternatives as well as the reasons why the current design was selected in favour of the rejected alternative (Selic, 2009).

If documentation is one of the tasks attached to the process for completing change request, then it is very likely that software engineers are going to update documentation (Lethbridge et al., 2003). However, if documentation is not attached to the process, then it is not certain that software engineers are going to update documentation (Lethbridge et al., 2003).

In accordance with the study from Lethbridge et al. (2003), more than half of the participants agreed that documentation is useful and effective for developers that are learning or working on a new software system. Furthermore, half of the participants found documentation to be effective when other developers are unavailable or when they are not looking for detailed or specific information individually. However, when confronted with the question of documentation effectiveness when maintaining a software system or working with an established software system, only about one-third of the participants answered that documentation is effective for both cases individually (Lethbridge et al., 2003).

Finally, in the study from Lethbridge et al. (2003), they talked with managers about the laziness of their employees, developers, to update documentation. Some managers answered that they try to impose a discipline on software engineers, meaning that they force them to update documents. However, Lethbridge et al. (2003) in contrast to many managers do not consider laziness being the problem, instead developers consciously or unconsciously make value judgements and conclude that it is valuable only to update certain types of documentation. Thus mitigation of the reluctance to update documentation should not be to force software engineers to do cost ineffective work but to endeavour for simple, but powerful documentation formats (Lethbridge et al., 2003).

2.3 Agile environment

This Master Thesis has focused on companies with an Agile environment. Therefore it is considered vital that the reader understand the basic principles of Agile development and how Agile methodology differ from traditional Waterfall methodology. Nevertheless, the differentiation in documentation practice between traditional Waterfall and Agile methodologies is not the aim for this Master Thesis.

Agile development is a general term for several iterative and incremental software development methodologies. Agile development fundamentally incorporates short iterations and continuous feedback that helps to refine and deliver a better software system. All while having a vision of continuous planning, testing and integration of the software. Also, Agile development is a way of organizing the development process, emphasizing direct and frequent communication – preferably face-to-face. Additionally, Agile development focus on customer collaboration in the development life-cycle and to make decisions together (Stålhane, 2008).

Agile software development recognizes that software development is iterative, exploratory and designed to facilitate learning as quickly and efficiently as possible. Nowadays the most common are Extreme Programming (XP) and Scrum (Nawaz and Malik, 2008). However other more lightweight approaches like Kanban has also gained popularity in the recent years. According to Abrahamsson et al. (2002), Agile methodologies like Scrum and XP are most suitable or at least most easily adoptable for small or medium sized companies because of the difficulties introduced in managing larger teams or synchronisation of many teams.

Scrum is an iterative and incremental Agile development method for managing product development within a team-based development environment. Scrum is the most popular and widely adopted Agile method because it is relatively simple to implement and addresses many of the management issues that have plagued IT development teams for decades (Nawaz and Malik, 2008). XP or Extreme Programming improves a software project in four essential ways: Simplicity, communication, courage and feedback (Nawaz and Malik, 2008). XP or Extreme Programming is a more

radical Agile methodology, focusing more on specific software engineering practices.

In order to illustrate the process here follows a description of the Scrum process (figure 2.1). Scrum start producing a product backlog that is a wish list of the customer's requirements (Schwaber and Sutherland, 2013). The development process consists of cycles called sprints which usually range from two to four weeks, where teams work to complete their tasks. The team allocate certain amount of time each day for a meeting where they assess its progress and discuss about their daily tasks, Daily Scrum. At the end of the sprint, the work should be ready for hand over to a customer, put in a store or show to a stakeholder. The sprint ends with a sprint review and retrospective meeting. During the sprint reviews, the Scrum team shows what they accomplished during the sprint. The retrospective meeting try to identify the things that the team has done well and that they should keep doing, as well as things they must start doing in order to improve in the next sprint. When the team chooses the next prioritized element of the product backlog and begins working again, then the next sprint has begun (figure 2.1) (Schwaber and Sutherland, 2013).

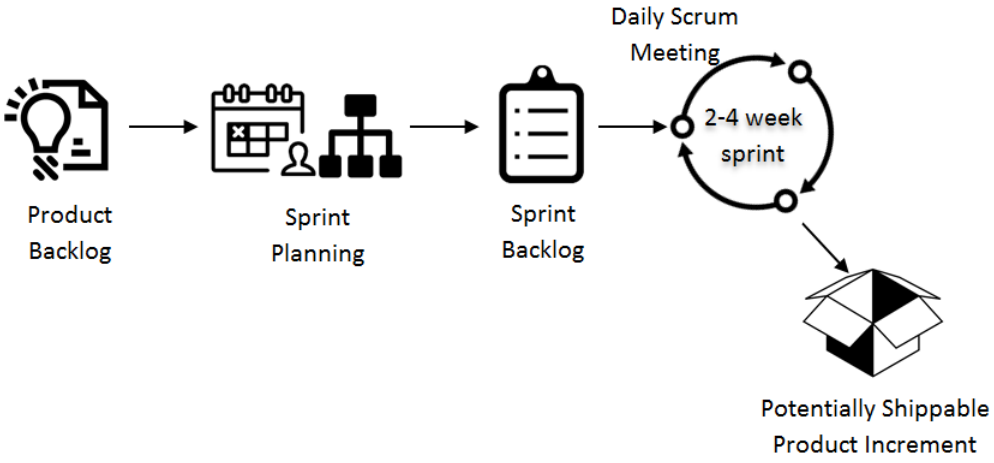
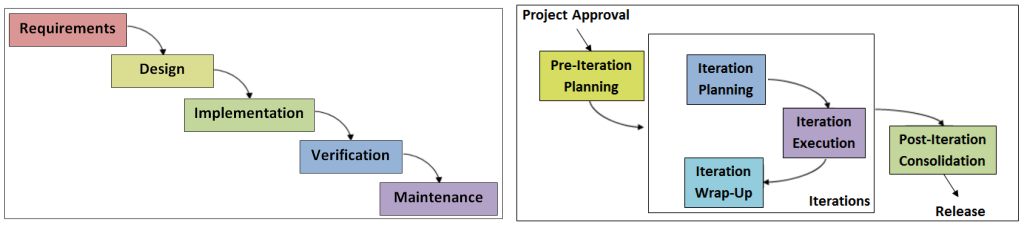


Figure 2.1: Agile software development based on Scrum (Adapted from Scrum Alliance, 2001)

Before Agile methodologies emerged most companies worked with waterfall approaches. One of the most important differences between the Agile and waterfall approaches is that waterfall features distinct stages with checkpoints and deliverables at each stage, while Agile processes have iterations rather than stages (figure 2.2) (Serena Software ,2007).
In Agile methodologies iterations are encompassed. Adopting Agile development processes, means to work in small teams, together with stakeholders to define quick prototypes, proof of concepts, or other visual means to describe the problem to be solved. The team defines the requirements for the iteration, develops the code, and defines and runs test scripts, and finally the users verify the results (figure 2.2) (Serena Software, 2007). These development iterations attempt to analyse, design, develop and test each feature and at the end of each iteration, the rest requirements are reprioritized. Additionally these iterations must be short, from two weeks to two months and finally a few weeks should be allocated to the delivery cycles (Larman,



(a) waterfall **(b)** agile

Figure 2.2: Waterfall process vs Agile process (Adapted from Serena Software, 2007)

2004).

Adopting Agile methodologies can be useful for the companies to achieve different objectives like: Communication, estimation skills, iteration planning and responsibility (Highsmith, 2004). Adopting Agile development practices, allows the company to achieve better quality on the product, reduction in lead times, cheaper systems and an increase in customer satisfaction (Abrahamsson, 2005).

2.4 Feedback

Additionally, this Master Thesis wants to emphasise the lack of feedback on documentation. Code reviews has also been reviewed, the researchers deem that the importance obtained for feedback on code are generally applicable to feedback on documentation as well. This is attributed to the inherent similarity of them both being a product of a text producing engineering activity, but also due to the similarities how they are received and communicated between software engineers.

The importance of taking continual feedback is the facility of learning from previous iterations and improve your performance in the future (Kamat, 2012). In the workplace the term feedback is used as the action of a review, to evaluate the performance of a person or group of people performing a task. As such, it is an action that reveals the strengths and weaknesses, the positive and negative aspects of a task, with the intention of improving it (Hattie and Timperley, 2007).

Feedback is a system of control over the development activities or tasks that are implemented through monitoring and continuously evaluating (figure 2.3) in order to gradually improve the output (Hattie and Timperley, 2007).

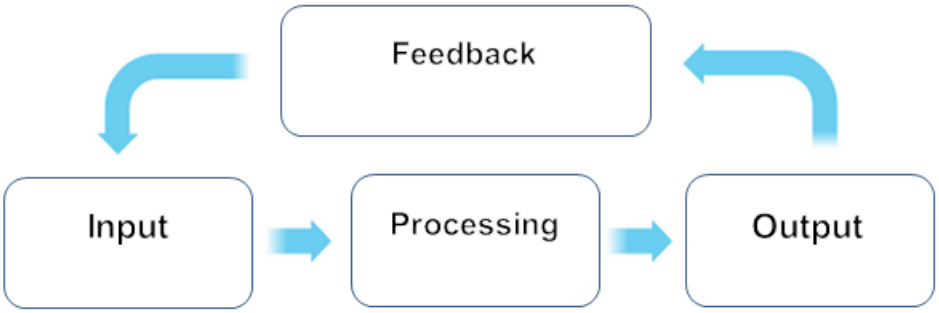


Figure 2.3: Feedback process diagram

Harry (2011), summarize the benefits of giving and/or receiving feedback as:

- Feedback helps you and your workmates to clarify your understanding.
- Feedback helps you see things in new ways (different point of views).
- Feedback helps you learn from your mistakes.
- Feedback makes you and your work better.

(Harry, 2011)

Although feedback has received little attention as a research topic for the Software Industry, it has been recognized by developers and managers as a significant factor in the different processes of software development (Meir et al., 1996). Feedback is the return on a particular subject or task (input) to a person or machine. Depending on the context, it may be the reaction, response or opinion. Hence, feedback can be both positive or negative in it's nature (Hattie and Timperley, 2007).

According to Robertson (2006) when your fellow employee is skilled in the task she has to review, then this is effective feedback because she will be able to provide a constructive feedback on the task. Receiving or giving continually constructive feedback to your co-workers can minimise the cost of a product by minimizing a product's development time. In addition it can also increase the likelihood of delivering a product satisfactory for the customer (Robertson, 2006).

Robertson (2006) suggested expanding the classification of the stakeholders of a project because classifying stakeholders in just developers or users was insufficient. To achieve the success of a project, a better classification of stakeholders had to be achieved. Figure 2.4, is a classification done by Robertson (2006) where internal stakeholders consists of Producers, Builders & Business and external stakeholders of Consumers, Sponsors, Influencers, Technical Consultants and Subject matter consultants (Robertson, 2006). The feedback will be dependent on the particular type of stakeholder and will depend on the knowledge and concerns prioritized by each stakeholder (Robertson, 2006).

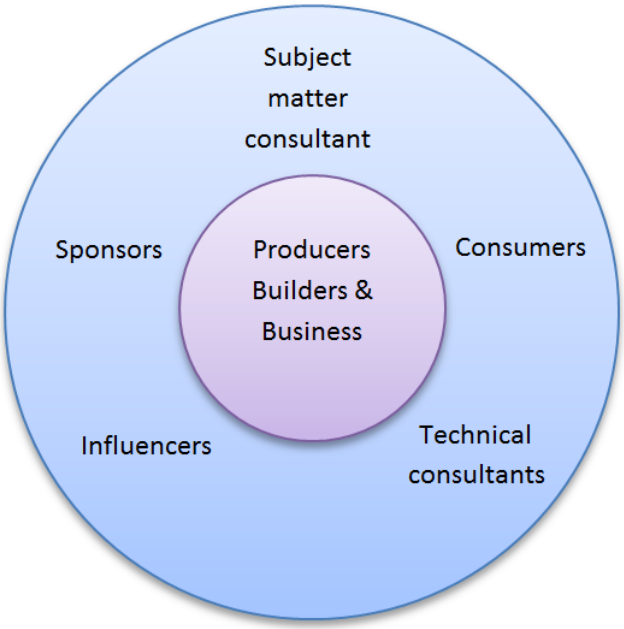


Figure 2.4: Stakeholders in a project (Adapted from Robertson, 2006)

In this study focus are on the feedback between internal stakeholders, and more precisely Producers. Producers are generally project managers, developers, designers and testers (Robertson, 2006).

The life cycle of a feedback loop will differ depending on if you conduct code inspection, documentation review or compile your source code (Ambler and Lines, 2012). The length of the feedback loop for both code and documentation may be days, weeks or even months in contrast to Test Driven Development (TDD) where the inherent feedback loops are completed within minutes. However focusing on Agile techniques the iterations in Agile practices reduce risks in programming and usually apply feedback faster than traditional software development methodologies.

Although adopting Agile practices do not guarantee anyone to solve all problems in the software industry, mostly due to factors uncontrollable by managers and their teams (Modigliani et al., 2014). Therefore Agile practices are not an assurance for the fulfilment of good feedback nor is the implementation of Agile practices necessary for a good feedback process. However it is usually beneficial for your feedback process to adopt an Agile work environment due to the strong inherent synergy between good feedback processes and Agile practices.

2.5 Prospect Theory

As the analysis investigates how changes in outcomes affects the perceived value of a task, prospect theory has been included to help the reader understand and facilitate an analysis about possible improvements.

The prospect theory is a theory created by Kahneman and Tversky (1979) (figure 2.5). It is a behavioural economic theory that describes the way people analyse a decision under conditions of risk but knowing the probability of each outcome. This decision is done in two phases (Kahneman and Tversky, 1979).

The first phase is editing, in this stage the person analyses the problem ordering the outcome of a decision using certain heuristics and set a reference point in order to clarify what is a gain and what is a loss. The reference point is really important because the outcome will be measured as a deviation from this reference point; the action to determine the reference point is called Coding. In the second phase, evaluation, the person makes a choice of the different options trying to maximize the value in his or her choice. Therefore a person will weight each outcome and multiply by the probability and he or she will select the one with highest value (Levy, 1992)

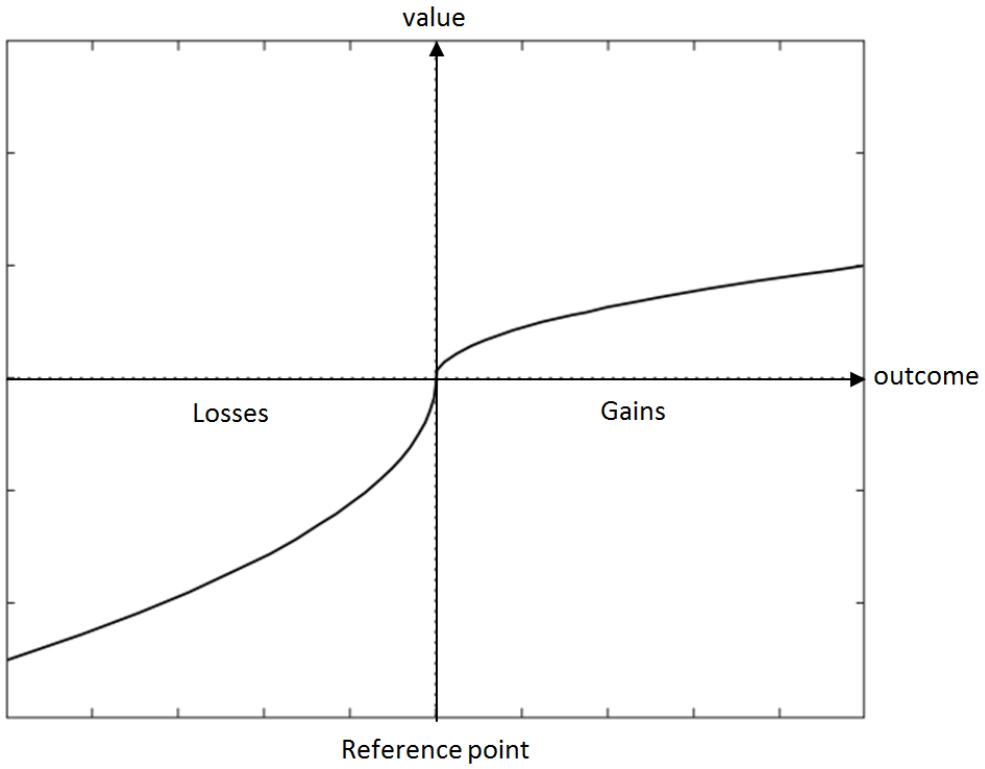


Figure 2.5: Value function in Prospect Theory (Adapted from Barberis, 2013)

The value function is shown in Figure 2.5 is defined in gains and losses rather than in terms of net assets, for that reason the choices are made from a reference point. Kahneman and Tversky (1979) argue that generally this reference point is the current situation or the existing condition (Kahneman and Tversky, 1979).

This S-shaped value function is steeper close the reference point and leads to an higher than proportional appreciation for gains close to the reference point. Nevertheless it is also appreciated that the function is asymmetric and is steeper for losses than for gains, this implies loss aversion (Kahneman and Tversky, 1979). According to Kahneman and Tversky (1992) most studies suggest that losses are twice

as powerful as gains, so for the same absolute positive (gains) or negative (losses) value, people assign higher value to avoid losses than acquire gains; for that reason, people tends to focus on avoiding losses rather than acquire gains (Kahneman and Tversky, 1992).
A consequence of an suboptimal documentation practice might be that software engineers strive to increase their documentation efforts. The researchers therefore deemed it important to include a theoretical baseline for how those improvements will be perceived and depend on the context in which they are undertaken.

2.6 Two selves and peak-end rule

Furthermore to understand of how experiences are remembered and evaluated the theory of two selves and peak-end rule has been included to describe how this process materialise. This is of particular interest since certain phases of the documentation practice will most likely have a bigger impact on the memory of said practice.

Kahneman (2011) describes how we choose between memories of experiences of experiences when we think we choose between experiences. Additionally we also visualise the future in terms of anticipated memories (Kahneman, 2011).

Kahneman (2011) believe that people confuse the words memory and experience. Kahneman (2011) affirms that people tend to remember short periods of intense happiness or peak moments at the end of an experience instead of longer periods of moderate joy (Kahneman, 2011).

In order to understand peak-end rule, Kahneman and Redelmeier (1996) decided to make an experiment tracking colonoscopy patients. The objective of this experiment was to find differences between the pain intensity experienced by the patients and the pain intensity that the patients thought they had experienced (Kahneman, 2011).

The figure 2.6 shows the graphic with the pain experimented from the two patients during the colonoscopy:

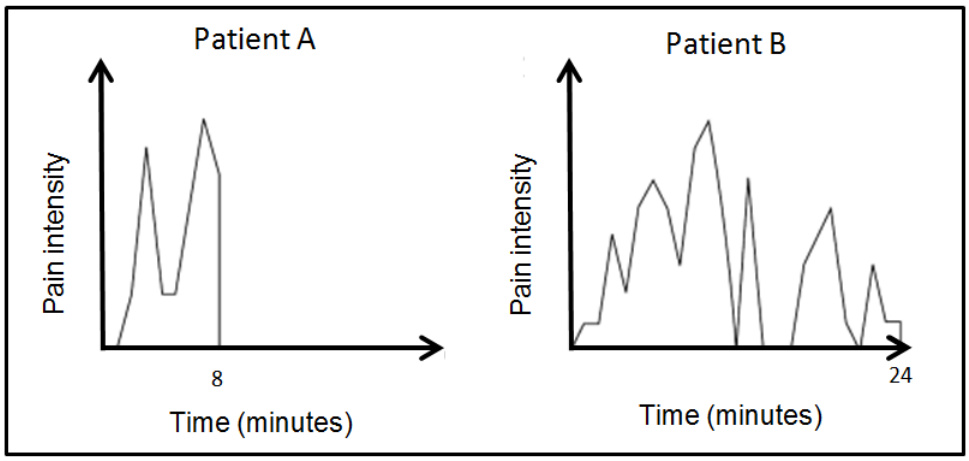


Figure 2.6: Experiences during the colonoscopy by patient A and patient B (Adapted from Kahneman, 2011)

The experiment procedure was the following: Each sixty seconds the patients had to note down the pain they experiment from a scale of one to ten. Redelmeier and Kahneman (1996) tracked 154 patients.

The experiment had a duration of eight minutes for the patient A, whereas for the patient B the duration was twenty-four minutes (figure 2.6). We can estimate based on the pain area for patient B that she suffered the most pain because the interval is much longer than for patient A. However, considering that both patients used the scale identically, Redelmeier and Kahneman (1996) found that the patient A thought she had suffered the most. The patient A had a worse memory from the colonoscopy compared to patient B. This is because the patient A remembers the pain as its peak at the end of the experiment. This difference is explained by the peak-end rule (Redelmeier and Kahneman, 1996) (Kahneman, 2011).

The peak-end rule illustrates the tendency that people have when they are evaluating experiences by focus on how the experience end and neglect the duration of the experience. The peak-end rule could then explain why patient A remembered the experienced worse although she suffered less overall (Kahneman, 2011). Is for that reason that although the patient B experienced more pain, the pain's area is bigger, she had a better overall memory of the experience. The explanation why she had a better memory than the patient A is that the pain was diminishing at the end of the experiment but also because the average intensity of peak was lower (Kahneman, 2011) (Redelmeier and Kahneman, 1996).

The two influencing aspects that the patients experienced from the experiment carried out by Redelmeier and Kahneman (1996):

1. **Peak- end rule:** The end and the peak moment of an experience play an important role in the general evaluation of the experience.
2. **Duration is neglected:** The length of an experience has marginal influence on the general evaluation of the experience.

(Kahneman, 2011)

Redelmeier and Kahneman (1996) pointed that in order to understand the happiness, we must differentiate between two selves. The confusion between those two selves is part of the problem of the notion of happiness:

- **Remembering self:** Is what we keep from the experiences. It is like our story. This is the self that keep score and maintains memories (Redelmeier and Kahneman, 1996)
- **Experiencing self:** Is the current mood of a person and how that person is feeling in the present situation (Redelmeier and Kahneman, 1996).

In comparison to the experiencing self, the remembering self is sufficiently stable and durable because memories is what people keep from their experiences (Redelmeier and Kahneman, 1996).

2.7 Job Characteristics Model

The Job Characteristics Model has been included in order to have a framework about job satisfaction for the interviews in order to facilitate a deeper analysis about primarily feedback.

For any company, job satisfaction of their employees is very important. A satisfied employee can be more efficient in her work, more productive and have better relationship with her colleagues (Hackman and Oldham, 1980).

Hackman and Oldham (1976) in order to explain job satisfaction developed the Job Characteristics Model. They used five job dimensions to describe the relationship between the job and what an individual expects to receive from his or her work. In Hackman and Oldham (1976) Job Characteristics Model (figure 2.7) there are five job dimensions that derive to three psychological states that influence to work outcomes.

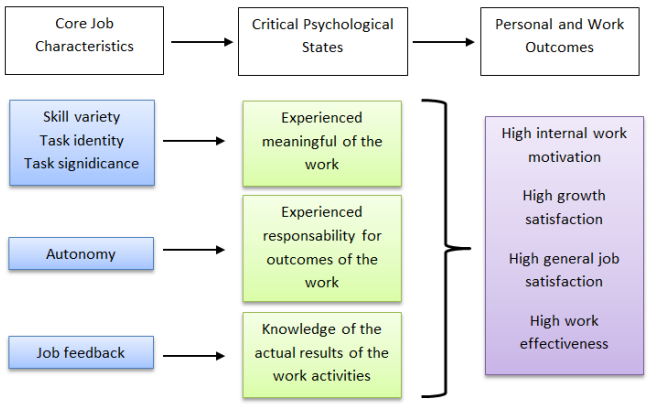


Figure 2.7: The job characteristics model (Adapted from Hackman and Oldham, 1976)

Hackman and Oldham (1980) define the five job characteristics as:

- **Skill variety:** Refers to the variety of different skills that an employee has to use in order to carry out different tasks in the workplace.
- **Task identity:** Refers to the degree of the employee's participation in all the tasks of the job from the beginning to the end of the production process.
- **Task significance:** Refers if the worker feels that the task she is doing is meaningful for the organization.
- **Autonomy:** Refers to the freedom or independence degree that the job gives to an employee when he or she has to schedule and carry out a task.
- **Feedback from job:** Refers if the employee has direct and clear information about how well the work is done. Feedback can be positive or negative, and good balances between both feedbacks are the most appropriate combination. Job feedback is a good way to improve their work performance.

As we can observe in the figure 2.7, skill variety, task identity and task significance are the three job characteristics that influence the most in the experienced meaningfulness of work. Additionally, the feeling of personal responsibility for outcomes of the work is derived from autonomy. Finally, the knowledge of one's work comes from job feedback.

All the job dimensions presented by Hackman and Oldham (1980) are important in order to achieve high internal work motivation, high growth satisfaction, high general job satisfaction and high work effectiveness. However, according to Hackman and Oldham (1980), the two job characteristics that has the most influence in order to achieve the highest motivating potential score (MPS) are autonomy and feedback (figure 2.8):

$$\text{Motivating Potential Score (MPS)} = \frac{(Skill\ Variety + Task\ Identity + Task\ Significance)}{3} \cdot Autonomy \cdot Feedback$$

Figure 2.8: Motivating Potential Score Formula (Hackman and Oldham, 1980)

2.8 Self-Determination Theory

A motivational theory part has been included in this chapter. Although Self-Determination theory is not exclusively related to the software industry it was considered indispensable to include. This motivational part has been one of the main supports for the analysis of the data. Self-Determination theory was by the researchers deemed as most applicable motivational theory.

Motivation is to have a goal, decide to reach out and be persistent in the effort to achieve it. The motivation is the stimuli which drives a person to perform certain actions and persist in them for completion. This term is related to the willingness and interest. Motivation in a workplace, is the will to make an effort to achieve the goals of the organization, conditioned by the effort's ability to satisfy some personal need. On the contrary an unmotivated person is one who feels no interest to perform

a task or activity (Ryan and Deci, 2000).

People can be motivated or unmotivated, nevertheless there are also different kinds of motivation. Therefore we do not just differentiate a motivated person based on her level of motivation, but also in her orientation of motivation. Orientation of motivation means to understand how a person is thinking, that is, the attitude a person has in order to achieve a goal (Ryan and Deci, 2000).

In Self-Determination Theory (SDT), a theory developed by Ryan and Deci (1985), they distinguish between different types of motivation according to the goals or reasons to perform a task or activity. Ryan and Deci (2000) report of studies demonstrating the differences between intrinsic and extrinsic motivation in terms of performance and creativity in the workplace, as well as the feeling of fulfilment by the employee in the workplace (Ryan and Deci, 2000).

Intrinsic motivation is the kind of motivation where a person does a task or an action by choice or self-interest, without expecting a reward, except for the gratification that the task or action inherently bring (Ryan and Deci, 2000). Intrinsic motivation can exist within individuals but usually exist in the relation between individuals and tasks, as not everyone is intrinsically motivated to perform a certain activity or task. For this reason, each individual has a unique set of particular activities or tasks that is intrinsically motivated to them (Ryan and Deci, 2000).

Cognitive Evaluation Theory (CET) was presented by Ryan and Deci (1985) as a sub-theory within the SDT that has the purpose of explaining the effects of social contexts on intrinsic motivation. The need for competence, autonomy and relatedness are satisfied by intrinsically motivated behaviours: (Ryan and Deci, 2000)

- **Competence:** According to deCharms (1968) the feelings of competence should go together with a sense of autonomy or by an internal perceived locus of causality (LOC) in order to enhance intrinsic motivation. Additionally, Ryan and Deci (2000) found that positive feedback plays an important role in fulfilling people's need for competence. That is, to increase people's intrinsic motivation, one way is giving people positive feedback on tasks as a positive reinforcement. On the contrary, negative feedback could diminish intrinsic motivation behaviour (Ryan and Deci, 2000).
- **Autonomy:** When you want to increase the commitment and motivation of an employee, giving employees more authority over their working lives can be a useful tactic. With autonomy, an employee has more responsibility and ownership of their work, which can motivate an employee to work harder and invest more energy and interest in the task or activity (Ryan and Deci, 2000). Deci and Vansteenkiste (2004) remarked that autonomy does not mean to be independent of other people, or in a workplace context, of your colleagues.
- **Relatedness:** Relatedness includes social and external esteem. Relatedness is the the desire to feel connected and experience caring for others like peers or other employees.

Intrinsic motivation is an important type of motivation and managers would like to believe that their employees are intrinsically motivated, nevertheless most of the tasks or activities people do are extrinsically motivated (Ryan and Deci, 2000).

Extrinsic motivation occurs when the individual's motivational stimuli are coming from outside. Therefore, the interest to perform a task is increased by external rewards such as money, promotion or a general sort of incentive. However extrinsic motivation can also be the recognition someone give us for our good work. Extrinsic motivation is needed to ensure that workers carry out tasks that are valuable for the firm (Ryan and Deci, 2000).

Everyone prefers to work without external pressure, to carry out tasks on their own and to value and self-regulate different tasks, that is, working intrinsically motivated. However, sometimes is not possible to mitigate this external pressure and fully transform the regulation into your own. The SDT use the term internalization and integration of value and behavioural regulation to explain this situation (figure 2.9) (Ryan and Deci, 1985 cited in Ryan and Deci, 2000).

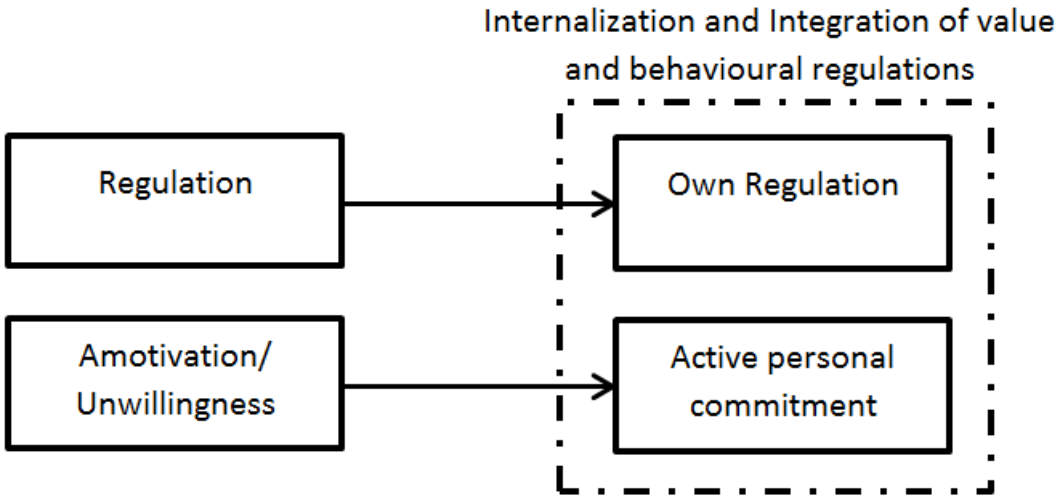


Figure 2.9: The process of Internalization and Integration of value

On the one hand, taking in a value or regulation is the process called internalization. On the other hand, when this regulation is transformed by employees into their own regulation, the process is called integration.

The benefits of internalization and integration of value and behavioural regulations are:

- Greater persistence in the tasks undertaken by the employees
- More positive self-perceptions
- Higher quality of engagement

(Ryan and Deci, 2000)

In the figure 2.10, a sub-theory to SDT proposed by Ryan and Deci (2000) called

Organismic Integration theory (OIT) is presented, which organizes several types of motivation according to the different degrees of autonomy or self-determination. The diagram shows the different types of motivation from amotivation to intrinsic motivation. In other words, the diagram goes from an impersonal (least autonomous type of motivation) to an internal perceived locus of causality.

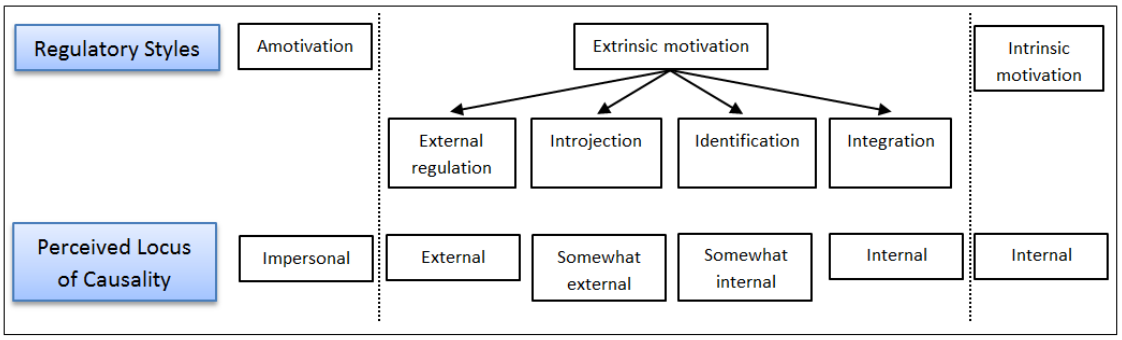


Figure 2.10: Organismic Integration theory (Adapted from Ryan and Deci, 2000)

Amotivation is the state of lacking an intention to act. This state occurs when a person feels lack of intentionality to perform an activity. Therefore the state of Amotivation arises when someone:

- Does not value a task or activity.
- Does not feel competent to perform the task or activity.
- Does not believe they can achieve the requested result.

(Ryan and Deci, 2000)

As it is showed in the figure 2.10 from Ryan and Deci (2000) four type of extrinsic motivation are distinguished:

1. **External regulation:** There is no or little perceived autonomy and the task is performed because someone asks us explicitly, or because we expect a tangible reward.
2. **Introjection:** The objective remains to fulfil an external demand, but the reward is internal. Introjected regulation is related to the maintenance of self-esteem, to rejoice in the ability to do something that someone ask of us. However introjected regulation does not emphasise the autonomy of the person.
3. **Identification:** The objective remains to fulfil an external demand, but in comparison to introjected regulation you have more autonomy and ability to make decisions. Here, the individual see the importance and the value of a task.
4. **Integration:** This type of extrinsic motivation is quite similar to the intrinsic motivation. In this case, the goal is to assure that the person assimilates the objectives as its own, in addition to having great autonomy. Even if they are similar it is important to distinguish Integrated regulation from intrinsic motivation because the task is not done for the inherent enjoyment of the task. However it achieves the best result of all the four types of extrinsic motivation.

(Ryan and Deci, 2000)

Ryan and Connell (1989) investigated if the four types of regulation presented by Ryan and Deci (2000) had a correlation. Ryan and Connell (1989) verified that it was an interrelation between the four types of regulation according to an ordered correlation pattern.

For that reason, depending on the type of regulation, people's attitudes will differ:

- **Externally regulated:** Is related to lack of interest, value or effort. Additionally, negative outcomes arise. Also there is a tendency to blame others, such as managers.
- **Introjection:** Is associated to anxiety and self worth, however is accustomed to invest effort into a task.
- **Identification:** Is related to a greater enjoyment than the previous regulations and the utility of a task has been aligned with one's selves values or standards.
- **Integration:** The regulations has been transformed and assimilated into your own. The individual usually show interest and enjoyment and the attitude is very similar to the one experienced with intrinsic motivation.
- **Intrinsic motivation:** Is associated with a great interest, enjoyment and feeling of competence for a task.

(Ryan and Connell, 1989 cited in Ryan and Deci, 2000)

2.9 Excitement and Value Model

Finally, an own theory has been construct by the researchers of this Master Thesis in order to provide a framework for understanding the data obtained from the interviews in this research study. This theory has a strong connection with the Self-Determination theory as well and has been used as additional support for the analysis.

However during the study, the researchers deemed the self-determination theory as not sufficient as a theoretical framework even though it was deemed to be the most appropriate motivational theory . This derives from the fact that the internalisation process is concentrated on one dimensional progress from more controlled to more assimilated regulations. The researchers had however at an early stage identified two distinct dimension for which progress could happen. Even if it required some review and investigation to construct the final version of the Excitement and Value Model it was at an early staged determined that a model with two independent dimensions would be necessary to facilitate an analysis of how motivation affect software engineers' behaviour.

The exact delivery in the software industry is often not clearly defined beforehand and the developer not only has to produce a solution but also many times define in more detail what problem to solve. This also consequently leaves the developer with

great autonomy of how to carry out the task. In contrast to the manufacturing business with an assembly line where the task is very well defined and the environment is consciously designed without autonomy of how to carry out the work. Given this comparison we can deduce it is part of the software developers work environment to decide and prioritize tasks. When deciding and prioritize tasks we hypothesize developers consider primarily two dimensions. How exciting a task is and how valuable a task is.

Ryan and Deci (2000) has shown in their research that the behaviour from someone intrinsically motivated is more confident, more persistent and exciting. It also affects the individual's behaviour with enhanced performance, higher self-esteem and more creativity. This was one of the supporting reasons to consider excitement as one of the prime motivational factors to be experienced in the workplace.

Nevertheless we can find employees lacking the inherent excitement for a task, instead motivated for extrinsic reasons; motivated for the separable outcome that the action will generate (Ryan and Deci, 2000). In this scenario the individual's inherent interest level to perform the task can range from disinterest to highly interested. We therefore consider a second dimension; the value perceived from and communicated to the employee. According to this dimension it is important for the employee to be acknowledged, mostly by his or her managers, that the task performed is valuable. The objective from the manager's perspective are thus to let their employees understand the importance and value they are producing for the company.

A task that fulfils the properties of both of those dimensions is most likely to be prioritized first. However if only the properties for one of those dimensions are fulfilled the value of the task seems to take precedence over the excitement. The researchers perception is that people actually work on tasks they consider valuable or is being appreciated by their bosses even if it is not exciting. However when the discrepancy in value or appreciation for the task disappears or are reduced to a marginal amount people will then instead try to choose the tasks what they find most exciting. If neither of the dimensions are met; the individual experience both low excitement and low value for the task the individual usually feel highly unmotivated and the task is most likely heavily deprioritized.

Here is a Excitement and Value Model (figure 2.11) with the concepts of value as the dimension on the first axis and excitement as the dimension on the other axis. We do not consider it to be only 4 distinct states but two continuous dimensions, however in order to simplify the presentation of the models properties. We have used the four most differentiating states that someone can experiment in a workplace (figure 2.11):

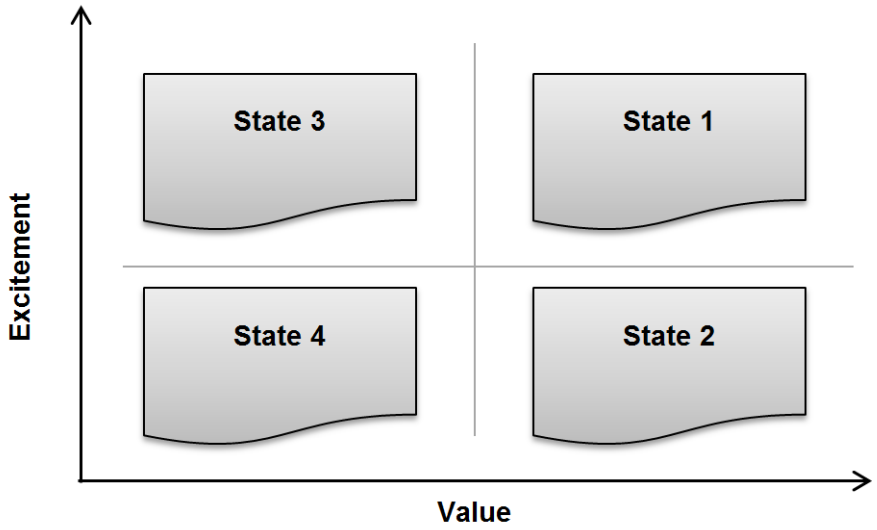


Figure 2.11: States of the Excitement & Value Model

The ideal state is state 1 because a task is highly valuable and exciting and state 4 is the considered the worst scenario. State 2 and state 3 are acceptable states in a short term perspective.

- **State 1:** When an employee enjoys doing a task, inherent excitement in the task is experienced. Additionally, the task is highly appreciated and valuable by the firm or your customers, peers or managers. Therefore the task is typically highly prioritized in your sprint backlog as well as motivating to be working on.
- **State 2:** In this case an employee feels a low inherent excitement in the particular task. He or she however considers the task to valuable or highly appreciated by the company, and therefore the task is usually prioritized. Although the activity is unexciting it still fairly prioritized and the employee is general very neutral about his or her emotions, due to being recognised as valuable for the company. So while state 2 short term is good from the company perspective and almost not noticeable worse than state 1, as well as decently stimulating to the employee. There is a risk that the task can find itself at the bottom of the priority queue only due to small changes to the task or introduction of a new task that triggers a reevaluation of the task's value. Leading to a number of state 3 task being prioritized before it as the employee's motivation is transitioned into state 4.
- **State 3:** In this case the employee also feels an inherent excitement for the task he or she is performing but it is not perceived as especially valuable. If we compare state 3 with state 1, the main difference is that in state 3 the task will not be prioritized in the product backlog because it is poorly appreciated by your managers. Hence, the task will be motivating to perform but generally not prioritized.
- **State 4:** In this case the employee is not excited doing a task and additionally the company do not acknowledge the task to be valuable. Thus the task is not

prioritized at all and it is highly demotivating.

3
Methods

This chapter has been divided into two sections. The first section introduces the theory regarding case studies this Master Thesis is based on. In addition it also introduces theory for how to analyse and collect data in qualitative studies. The second section is the Research design where the methodology and procedures used during the the study is described.

3.1 Case Study

A case study research method is an empirical analysis that investigates a contemporary phenomenon within its real life context (Yin, 1984). Therefore a case study is a detailed research study of a specific situation and it is basically used to narrow down a very extensive field of study into one easily researchable topic (Shuttleworth, 2008). Although using case studies do not guarantee to answer the research questions, it gives you some indicators and allows you to provide hypothesis on a topic (Shuttleworth, 2008).

Different data sources can be used in case studies such as surveys, interviews, observation and interaction of people both at qualitative and quantitative studies (Yin, 2004) (Eisenhardt, 1989). In order to increase the quality of data collection and the originality and brilliance of the data analysis process, Eisenhardt (1989) suggests that various researchers should work together when managing case studies.

According to Yin (2003a cited in Baxter and Jack, 2008) a case study must be considered when:

- The types of questions 'how' and 'why' are investigated in the research.
- The behaviour of the people involved in the study is difficult to manipulate.
- Contextual conditions are covered due to being relevant to the phenomenon studied.
- The boundaries are unclear between the phenomenon and the actual context.

The selection of a specific type of case study model will depend on the study purpose; that is your decision will be influenced if you want to describe a case, to explore a case or to compare two or more cases. There are two authors that describe a variety of case studies using different terms. Yin (2003a) and Runeson and Höst (2009) categorizes case studies as explanatory, exploratory, or descriptive. For all three categories Yin (2003b) also distinguishes between single, holistic and multiple-case studies; whereas Runeson and Höst (2009) differentiate between single-case study

design and multiple-case study design. Stake (1995), in contrast to Yin (2003b) and Runeson and Höst (2009), identifies case studies as intrinsic, instrumental, or collective.

On the one hand single-case studies are an inquiry of one individual or a group. On the other hand multiple-case studies use the process of replication which consists of choosing cases with similar results (NCTI, 2012), that is the goal is to replicate findings across cases (Yin, 2003a).

In the figure 3.1 the four types of designs for case studies is showed:

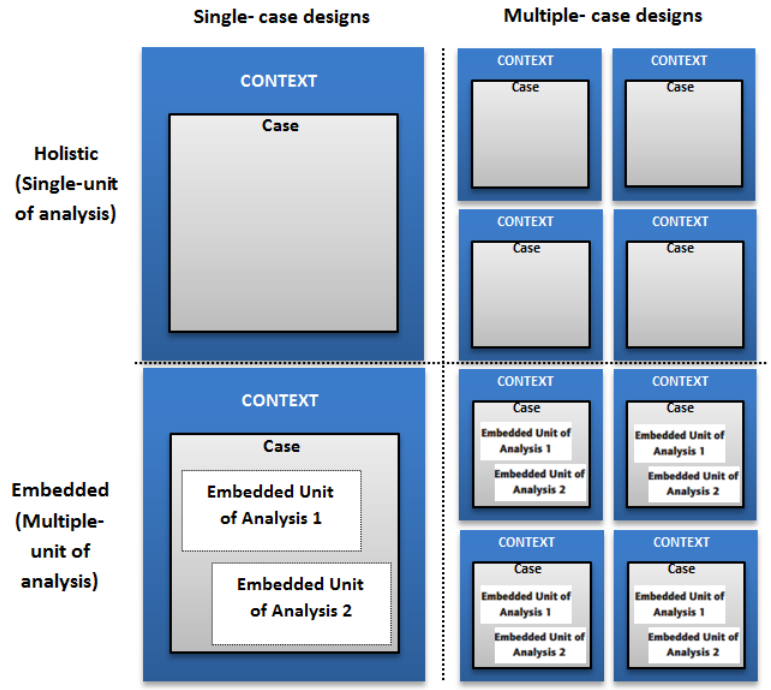


Figure 3.1: Four basic types of designs for case studies (Adapted from Yin, 2014)

- **Single-case (holistic) designs:** The case study examine the unique environment or context as one single unit of analysis.
- **Single-case (embedded) designs:** The case study examined involve more than one unit of analysis within one unique environment or context.
- **Multiple-case (holistic) designs:** The same study may contain more than a single case. In this type of design, each cases are considered its own unit of analysis and several cases are examined to understand the similarities and differences between them.
- **Multiple-case (embedded) designs:** The same study may contain more than a single case. In this case multiple units of analysis are examined within each context, but the data obtained in each unit of analysis will only be analysed within that given case. Therefore a multiple case design with embedded units only allows the researcher to focus one unique or critical case.

(Yin, 2014)

The exploratory case study is the most common alternative (Runeson and Höst, 2009) and is used to investigate a relatively new or unexplored field of scientific investigation in which the research questions have not been clearly identified and documented. The exploratory case study is very often applied as a preliminary step of a descriptive or explanatory case study research (Albert et al., 2009) (Runeson and Höst, 2009). As Yin (2003a) mentioned in his list of factors, selecting a case study strategy, an exploratory case study investigate the 'how' and 'why' questions of a phenomenon but not the 'how many' or 'how much'. The last two type of questions are instead characteristic of surveys or archival analysis (Yin 2004) (Runeson and Höst, 2009).

Many well-known case study researchers (such as Stake(1995) and Yin (2014)) have written about case study research and proposed techniques for organizing and conducting the research successfully and they proposed six steps when conducting a case study:

1. Define the research questions
2. Determine data collection and data analysis techniques
3. Prepare to collect the data
4. Collect the data
5. Analyse the data
6. Prepare the report

(Harmon, 1997)

The main benefit of conducting a case study lies in the particular details and holistic understanding researchers gain from a specific case. Case studies allow researchers to perfectly understand how selected individuals conduct their thought process and why certain actions or objectives have an effect in a specific context (NCTI, 2012).

3.1.1 Literature Review

The first step to do once the topic of the case study research is identified, is to investigate what is already published and what is not about the selected topic. Analysing existing literature is a good practice to identify essential research questions or hypothesis. Eisenhardt (1989) states that one of the benefits of theory building research is the corroboration of new hypothesis with existing literature (Eisenhardt, 1989).

The aim of a literature review is to ensure the importance of your research question or questions and to analyse what research models or designs are most suitable for your study (Hancock and Algozzine, 2006).

On the one hand Hancock and Algozzine (2006) affirms that reading studies and projects of others can teach you formats and procedures for communicating your findings and conclusions. On the other hand Eisenhardt (1989) says that the benefit of consulting research is to determine contradictions and similarities between your findings and the consulted research. Furthermore also to provide a starting

point for the analysis where the aim is to investigate why conclusions are different or similar (Eisenhardt, 1989).

Doing case study research means determining what it is known about a matter, to establish its importance and the need for further exploration using the research questions (Hancock and Algozzine, 2006).

3.1.2 Determining the case/unit of analysis

The unit of analysis is the *who* or the *what* that is analysed in the study. In order to provide generalisability for the results of the study the unit of analysis is used in to determine the level of abstraction. Moreover your unit of analysis can be an individual, a group or a program (Trochim, 2006). However the most common unit of analysis is the individual because it is the level at which we often synthesize and compare data (Guest et al., 2013).

Sometimes the unit of analysis can be the case, and then we would say that the case study is holistic. In contrast, if the unit of analysis are defined as subunits of the case, we would define the case study as embedded (Yin, 2004). An embedded case study may treat companies as cases and individuals as units of analysis, for instance.

As mentioned above, the case can be a group, additionally the case can also be an individual, an organization or some larger unit. In software engineering research the case are usually software engineers, teams, software systems or development processes (Runeson and Höst, 2009).

3.1.3 Data Collection

There are several different types of interview approaches. In qualitative studies semi-structured interviews are the most common approach. There are also structured interviews that are more appropriate for a quantitative research method and finally unstructured interviews is another option, although very time-consuming, it can be used for both quantitative and qualitative studies (Seaman 2008). One of the differences between a semi-structured interview and structured interview is that in a semi-structured interview you have planned questions but can ask them in a random order. Furthermore in a semi-structured interview you are allowed to ask questions that are not written in the interview template, giving you the possibility to investigate further about an interesting topic (Runeson and Höst, 2009).

Additionally, interviews are preferably recorded in order to get the maximum value out of information from interviews. Especially considering the fact that quantity of data is an important factor to support the quality of the discussion and conclusion essential in providing a solid qualitative study. According to Nastasi (1999), a ten to twenty hours database should be enough to support a qualitative study as illustrated in (Table 3.1).

Number of Interviews	Length of each interview
10	1 – 2 hours
20	30 minutes – 1 hour
30	20 – 40 minutes

Table 3.1: Length of Interviews

3.1.4 Data Analysis

One of the most difficult things of doing case studies is the data analysis. For that reason Yin (2003b) describes five techniques for analysis:

1. **Pattern matching:** In case study analysis, one of the most acceptable techniques is to use a pattern-matching logic. This technique consists on finding patterns and then constructs an interpretation of those patterns.
2. **Explanation building:** You can build an explanation about the case analysing the case study data.
3. **Time series analysis:** The ability to trace changes over time
4. **Logic models:** Consists of matching empirically observed events to theoretically predicted events. It seems to be a sort of pattern matching but one of the main differences is the use of sequential stages in the case of logic models, therefore we must consider these techniques differently.
5. **Cross case analysis**

Yin (2003b, p. 137) argues that "no matter what specific analytic strategy or techniques have been chosen, the researcher must do everything to make sure that the analysis is of the highest quality." A qualitative analysis is no excuse to be less careful and should be attempted with the same precision as an quantitative study.

In contrast to Yin (2003b), Stake (1995) describes two different type of analysis:

- Categorical aggregation
- Direct interpretation

Finally, Eisenhardt (1989) identifies:

1. **Within-case analysis:** The aim of within-case analysis is to dig deeper on the phenomenon under study. Within-case analyses are useful within a single case. The intention of this analysis is to get familiar with each case individually and then starting to generalize patterns traversing cases (Eisenhardt, 1989).

2. **Cross-case analysis:** Searches for patterns between cases (Eisenhardt, 1989). Using a cross-case analysis is to simplify the comparison between cases and identify similarities and differences (Eisenhardt, 1989). Another tactic is to choose different categories. Stratify the findings according to the category chosen can help to identify patterns. These categories can be for example size of the company in the case, or business field of the company (Eisenhardt, 1989).

In a case study, data analysis each data source and the findings extracted from them cannot be treated separately. Moreover, you must ensure that individual pieces of information not deviates too much from the overall picture, in an attempt to understand the case as a whole. A good tactic to ensure that you are considering the big picture of the case is to involve your co-worker in the analysis phase (Baxter and Jack, 2008). In this study involving both Master Thesis researchers in the data analysis can ensure that the incorporation of data sources answers the research questions.

3.2 Research Design

This study has been carried out as a holistic multiple case study (Yin 2004). This study aimed to investigate how and why developers are reluctant to produce technical documentation. According to the fact that this topic is mainly unexplored, the research of this project focuses on practitioner's perceptions. Considering that a descriptive or explanatory research was unsuitable, the Master Thesis has been developed as an exploratory study. This exploratory case study examined how inhibitors to write technical documentation is manifesting and aimed at generating hypotheses to explain the phenomena.

In order to answer the research questions, we have produced an interview template to inquiry the thoughts of software developers, architects and project/product managers. Our approach was to verify the from literature constructed hypotheses based on the information we get in the interviews about software documentation and inhibitors in IT firms. Additionally, using a symmetrical approach, we generated new hypotheses from the information extracted in the interviews and compared and verified that with existing literature.

3.2.1 Pilot study

Once the research questions and questions have been defined by the researchers, is time to try out those questions applying techniques and methods which have been applied during the entire project (Blaxter et al., 2010). The usefulness of a pilot study is to test research techniques and methods that the researchers have in mind to see how well they will work in practice. If necessary it can then still be adapted and modified accordingly (Blaxter et al., 2010).

The value of a pilot study is high, and according to Blaxter et al. (2010, p. 138) states that: "You may think that you know well enough what you are doing, but the value of pilot research cannot be overestimated. Things never work quite the way you envisage, even if you have done them many times before, and they have a nasty habit of turning out very differently than you expected". It is for that reason that conducting a pilot study could help to prevent waste of time, energy and money for the researchers. Furthermore, Welman and Kruger (1999) list three advantages of conducting a pilot study:

- A pilot study helps to detect possible defects in measurement procedures such as time limits.
- A pilot study is also valuable to identify unclear or ambiguous questions.
- A pilot study helps to recognize the non verbal behaviour of participants. It is possible to detect questions that are embarrassing or upsetting for the interviewees, due to a bad use of the words or because the form of the question is out of order.

Before conducting the main interview, a pilot study was conducted to help develop and refine the questions. The pilot-study participant was a degreed Software Engineering Master student of Chalmers University of Technology with one year and a half of experience in software development.

The pilot study allowed the possibility to analyse the answers and tailor the interview process to better match the research questions in this study. The definitive interview template, created in March 2015, with more concise questions and deep emphasis on feedback, was used with all participants.

3.2.2 Selection of question

The semi-structured interviews consisted of a wide range of questions, in total amounting to thirty-eight questions. It ranges from questions related to the practices and guidelines used in the company to questions about the developers', architects' and project managers' opinion and motivation about practices as well as work habits.

Although all questions of the template are not asked, the main purpose of the template is to help the researchers to cover all the relevant topics, without losing the mental model/ framework. All the question asked are neither included in the template as it is not viable to design the template to accommodate and predict when the interviewers will find it useful to ask follow up questions. Usually in order to explore a specific topic deeper due to the answer of the interviewee.

The questions can be categorized into:
- Personal information of the interviewee
- Company practices and guidelines
- Personal opinion

3.2.3 Participants

During this project, the researchers decided that the best way to carry out this Master Thesis was by doing a qualitative research because it was the most suitable for the analysis of data deduced from the interviews. The data that is used in this qualitative research study is collected from interviews with IT companies in Gothenburg. We also requested that all interviewees had at least half a year of experience in the software industry, nevertheless several had over twenty years of experience.

Discovering and understanding the experiences, points of view and thoughts of par-

ticipants was one of the purpose of a qualitative research methods (Hiatt, 1986). The purpose of using a qualitative research method was because it allowed a detailed examination of a topic in which the information is collected by a researcher using interviews and case studies (Harwell, 2011).

It is possible to obtain different results from the same interviewee depending on who the researcher or the interviewer is. This explanation fall on that people have different skills regarding critical thinking, problem solving, initiative and handling data (Showman et al., 2013). If we focus on qualitative studies, a good qualitative researcher ask questions, then listens, then thinks and finally asks more probing questions to get to deeper levels of conversation (Simon, 2011). It is known that people have different skills or a different degree of them, and therefore divergent virtues. In a flexible and open research process, the interaction between the participant and the researcher is an inherent property of the interview process and it has been taken into consideration during the analysis of the data (Harwell, 2011).

Using this inductive method (qualitative research method), the data provided by a participant has been used in order to construct theories or hypotheses (Harwell, 2011) and afterwards compared with the existing literature.

3.2.4 Demographics

This section describes the participants' demographics. The divisions separate individuals based on software experience, current company size and software duties. The purpose of this section is to show that the interviews conducted were broad and therefore possibly can be extended to other software contexts. Even if the generalisability was the primary objective in the study, the variety of people interviewed could enhance this study's generalisability.

The figure 3.2 illustrates the participant's experience, based on numbers of years in the industry:

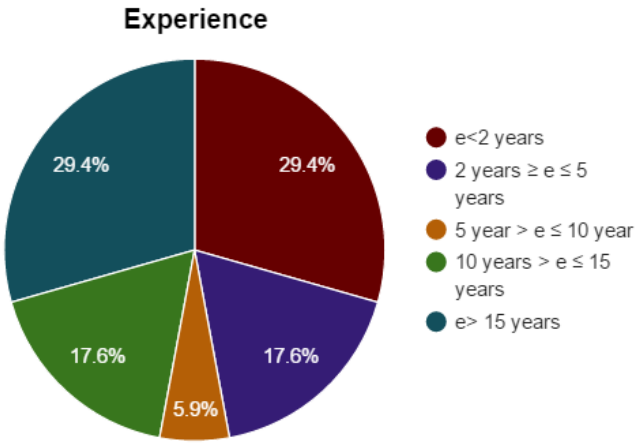


Figure 3.2: Participant's experience

The figure 3.3 illustrates the size (micro, small, medium or large) of each one of all the companies studied according to SME (2005) (table 3.2). Where a company qualifies for a category if they does not exceed both the upper limit for annual work unit and either the annual turnover or annual balance.

Company category	Annual Work Unit	Annual turnover	Annual balance
Large	≥ 250	> 50	> 43
Medium	< 250	≤ 50	≤ 43
Small	< 50	≤ 10	≤ 10
Micro	< 10	≤ 2	≤ 2

Table 3.2: Company category (Adapted from SME, 2005)

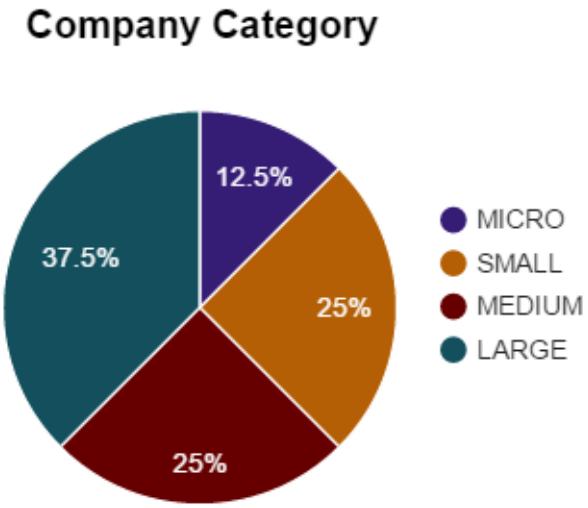


Figure 3.3: Size of each one of all the companies

The figure 3.4 illustrates the current job functions (at the time the interviews were conducted) by the participants:

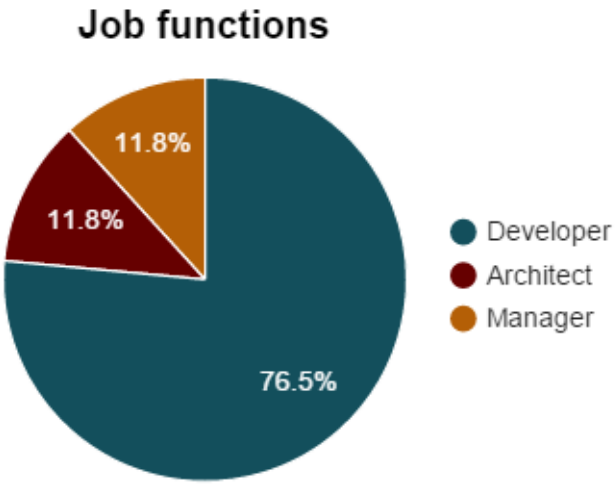


Figure 3.4: Current job functions

3.2.5 Collecting data

A data collection method in form of semi-structured interviews has been used. All interviews were recorded, and also transcribed literally, although transcribing is very time consuming, estimating that a one hour interview takes around six hours to transcribe (Singer et al., 2008). Not transcribing the interviews may trigger to a loss of that information (Singer et al., 2008).

The interviewees were aware about the confidentiality of the interviews. Additionally, in the report, coding was used in order to hide the identity of the interviewees as well as the name of the company.

In this Master Thesis, seventeen interviews were carried out with a length between thirty-five minutes to one hour for each interview. According to the Guideline for Length of Interviews provided by Nastasi (1999) (see Table 3.1) this seemed to be enough to back the data up.

3.2.6 Analysing data

The purpose of analysing data was to obtain usable and useful information. Qualitative research is sometimes criticized by quantitative researchers to be too subjective, but this was not the only concern raised when carrying out a qualitative study, quantitative researchers further argue that these studies are difficult to replicate because the study is product based on the researcher's predictions (Bryman and Bell, 2011).

In order to mitigate the risk of personal bias and assure the quality of the qualitative research, reliability and validity have been used as measures to evaluate that

quality. The reliability and validity measurements can then be split up into internal and external reliability as well as internal and external validity.

External reliability of a qualitative study refers to the degree of duplicability of that study (Bryman and Bell, 2011). According to Goetz and LeCompte (1984), they affirm that the research results are determined by the social setting and circumstances of the initial study. However, the researchers were concerned about this and from the beginning of the study they decided to create an interview template in order for future researchers to have a greater possibility to replicate the study under different circumstances.

Internal reliability arise when in a research there is more than one observer or researcher, and therefore they have to analyse and collect the data together and agreeing about what they see and hear (Bryman and Bell, 2011). This is an important aspect in qualitative research and the issue with multiple researchers have been minimized due to the agreement between the two researchers to attend the semi-structured interviews together. Also a subsequent transcript of all interviews helped out to better analyse all the data obtained. Furthermore after each interview the researchers found it a worthwhile activity, to discuss and analyse the answers in order to align the researchers opinions and interpretations about key answers.

Internal validity refers to the level of congruence between the empirical observations and the concepts derived from them (Bryman and Bell, 2011). In order to ensure a high degree of internal validity, triangulation has been used. The purpose of triangulation in qualitative research is to improve the validity of the conclusions. In order to shed light on a phenomenon, using multiple data sources is more adequate to ease the understanding of the issue (Rothbauer, 2008).

Then triangulation is defined to be "a validity procedure where researchers search for convergence among multiple and different sources of information to form themes or categories in a study" (Creswell and Miller, 2000, p. 126). Triangulation is a well established method for researchers to avoid relying too much on a single source.

Denzin (1978) and Patton (1999) identified four types of triangulation:

- **Across data sources:** Collecting data at different times, social situations and to different people.
- **Theories:** More than one theoretical position in interpreting data.
- **Methods:** More than one method for gathering data; i.e interview, observations, documents.
- **Among investigator triangulation:** More than one researcher in the field to extract and interpret data.

However, the researchers considered triangulation of sources and investigator triangulation the most feasible for this research. Firstly, triangulation of sources consisted of examining data manifesting as a patterns across several different data sources from the semi-structured interviews. Secondly, triangulation among investigators consisted of both researchers inspecting, examining and analysing the data obtained

from the interviews.

External validity refers to the degree to which the outcomes of the study can be generalized across different social settings and typically affects qualitative research due to their often small sample sizes (Bryman and Bell, 2011). The fact that the researchers found interesting to do more than one interview in some of the companies they interviewed was to also facilitate an analysis where you identify patterns between people in the same organisational setting. However the researchers were aware about possible selection bias in those specific companies. Moreover, the variation between the number of participants within each firm is relatively small (one to four interviewees per firm) and each interview was also carried out individually.

One of the issues identified by the researchers was the sample size, however in qualitative studies you do not have strict rules for the minimum and/or maximum number of interviews. That does not mean however that the sample can or should be arbitrary large, instead focus can be on how large does the sample size have to be in order to obtain consistent patterns. Although there is no definitive answer about the sample size there are methods to help assure that the sample size is large enough. What can be done is that after the first ten interviews you analyse the data collected and you determine if information begins to be redundant or if there are new concepts emerging. If new concepts are emerging you should continue the data collection until saturation occurs (Nastasi, 1999). Saturation occurs when data is being collected and no new or relevant data shed light on the issue that is being investigated (Given, 2008). Patton (2001) proposes a qualitative sample strategy, homogeneous sampling, claiming that bringing people of similar backgrounds and experience helps to reduce variation and facilitate analysis of data.

4
Results

The data presented in this chapter has been collected from seventeen developers, architects and project leaders at software firms in Gothenburg through interviews. In this chapter the findings of each case will be presented. The study is a qualitative study and therefore focus is primarily to understand the reasoning behind rather than quantify certain trends.

The Table 4.1 shows a summary of all cases:

Name	**Company**	**Role**	**Experience**
A1	Alfa	Project manager and team leader	22 years
A2	Alfa	Chief Architect	26 years
A3	Alfa	Developer	1,5 years
A4	Alfa	Developer	15 years
B1	Beta	Developer and Scrum Master	12 years
B2	Beta	Test architect	7 years
B3	Beta	Developer and Scrum Master	3 years
B4	Beta	Developer	1,5 years
C1	Gamma	Developer	0,5 years
D1	Delta	Business area and project manager	37 years
E1	Epsilon	Developer	5 years
F1	Zeta	Developer	1,5 years
F2	Zeta	Developer	14 years
G1	Lambda	Developer	1,5 years
H1	Omega	Developer	3 years
H2	Omega	Developer and Scrum Master	17 years
H3	Omega	Developer	20 years

Table 4.1: Summary of interviewees

4.1 The cases

Each case is presented with a short summary, findings and key findings.

4.1.1 A1, Alpha

A1 is a project manager and team leader in the company Alpha. She has over twenty-two years of experience from software development and although she is no longer a developer herself she has daily contact with the developers at the firm and also has experience as a developer from earlier in her career. At Alpha they follow an Agile philosophy where they have customised and picked what they consider to be the best part from Scrum.

Findings

A1 recognises a lack of enthusiasm from the developers towards documentation and that developers do not seem to be equally motivated by the documentation activity and the coding activity.

She mentions that they are a consultancy firm and that their primary income is what they get paid from the customer. The customer in turn expects them to deliver code as that is what they consider to be the working product. A1 mentions that because their income is so heavily tied to the request from the customer, that dictates what tasks Alpha decide to engage in.

Nevertheless she recognises that it will be easier to maintain the code if you have documentation and emphasises it is important to understand that you make a commitment to keep it updated. A1 thinks that documentation that is not updated is misleading. She proposes that a good way to mitigate the problem of outdated documentation could be using a tool that helps the developer to write documentation. When confronted of why this lack of motivation she becomes a bit uncertain but feel that lack of structured guidelines and instant feedback might be an explanation of this shortcoming.

Furthermore she understands that the tight deadlines might be inhibitive for documentation activities to flourish and that a new view including more documentation would need to be enforced from the company. A1 also mentions that some developers at Alpha believe that code should be self- explanatory and therefore you do not need documentation. If more documentation is to be adopted in this firm she believes that the firm has to enforce it since most people see enough value of the documentation themselves.

When A1 was asked about feedback on documentation, she admitted that they do not have it. However at Alpha they have code reviews and something called, code afternoons, where the architects and developers sit down and show good or bad examples of code that they had written, so in that sessions there is a bit of a feedback on things that they have done.

Key Findings

- The developers do not like to write documentation the same as they do for writing code.
- Documentation not up to date might be harmful however updated documentation facilitates the maintenance of the code.
- Due to the criticality of accurate documentation a way forward could be more automatic tools that would take a bigger burden of the documentation responsibility.
- If more documentation is to be adopted in this firm she believes that the firm has to enforce it since most people see enough value of the documentation themselves.
- There is usually no feedback around documentation as this activity is not requested by the customer.

4.1.2 A2, Alpha

A2 is a chief architect in the company Alpha. He has over twenty-six years of experience from software development and among his more common tasks are designing and communicating the architecture of the system but also code review and providing feedback to the developers. At Alpha they follow an Agile philosophy where they have customised and picked what they consider to be the best part from Scrum.

Findings

A2 has an interesting standpoint on documentation from his role as an architect. He firmly believes that documentation has to be very lightweight as it is very costly to keep updated and that comments in general are superfluous. He does provide some further explanation about this and that is that comments should be provided when you are doing some sort of hacking and depart from normal coding conventions or standards. People do not really expect to get any feedback on documentation.Additionally, people are not really interested in receive feedback on documentation.

He does moreover also provide an interesting perspective on the change that has happened in software engineering over the last couple of years, where the speed at which software is developed has significantly increased. In his view and from his own experience this has also changed the role of documentation, before when the speed at which software documentation could be produced it made sense to more carefully document everything. To make sure that code was able to be maintained and developers able to manage changes of the system's purpose. However today code is produced at a much higher speed, the system's purpose change much faster and to a greater extent than ten years ago according to A2 and it is not really feasible to keep everything documented at the level of detail that was done before.

A2 also describe how tools have become more sophisticated and things like traceability can now be provided from the developing environment rather the documentation.

Overall A2 feels that a lightweight documentation approach might be the right trade-off and where return on the investment can be achieved for documentation. A2 also point out that it is time saving with documentation when introducing new people into a project and that fact could increase the return on investment.

He does however seem to contradict himself when saying that they could produce good documentation if they changed their routines, at the same times stating that documentation would not help up to clean up the mess that forces them to redesign the system ever so often. A2 recognise however how especially lightweight documentation can be very beneficial in redesigning systems. The only reason of not producing this beneficial documentation seems to be that the customer does not want to pay for it.

Key Findings

- A2 points out while customers like to have documentation they are not actually willing to pay for it.
- It is very important to have documentation updated, as out of date documentation is potential harmful. Although it is very costly to have documentation updated.
- Documentation has the most potential when introducing new people into a project or area.
- People do not expect or interested to get any feedback on documentation
- Some tweaks to their routine could help to produce good lightweight documentation which could be potentially beneficial.

4.1.3 A3, Alpha

A3 is a developer in the company Alpha. He has around one and a half year of experience as a software developer. His daily tasks could include anything from writing to producing tests or other artefacts needed. A3 has also taken a greater responsibility towards architecture. As for the other employees in Alpha they follow an Agile philosophy where they have customised and picked what they consider to be the best part from Scrum.

Findings

A3 in his role as architect produces suggested solutions, which could be seen as a form of documentation; however he is very much against calling it documentation. He does find some benefit of not having to repeat himself, however the request for the type of documentation that he could considered useful, the why, is very low. When faced with question about it, he considers it as something useful but as people generally do not need nor request it, also no value in producing it. A3 considers that you do not need documentation for other peoples code, because you just have to look at the code and documentation is potentially misleading in that aspect.

An interesting view is that A3 does not consider there to be much value in documentation unless it is documentation for external code. He basically does not want documentation on the same level as the code. This sort of generic documentation that he describes is in his opinion actually useless. In the cases he does document his code it is usually because he been forced to do some hack or implement what he consider stupid, usually due to the customer not interested in a dynamic solution and want something hard coded into the system. Furthermore A3 emphasis it also depends on if it affects something somewhere else in an unexpected way, then he is inclined to document it, but if it local to the function he feels you do not really have to anyway. A3 believes that documentation for application programming interfaces (APIs) is good and that is something he considers part of a good standard. However he does consider this entirely different as this is external code base and in many cases he does not have access to the code itself, or it would be too time consuming to consult the code.

A3 does not think that lack of feedback is really an inhibitor for not producing documentation. He has the opinion that feedback about the process would help much more than feedback about the content. At Alpha he rarely receives any direct feedback on documentation and he does not expect feedback.

Key Findings

- Code should be self-explanatory, therefore he does not document his code.
- In the cases he do document his code it is usually because he been forced to do some hack or implement.
- Believes that documentation for APIs is good and that is something he consider part of a good standard.
- He considers that documentation is something useful but as people generally do not need nor request it, then there is no value in producing it.
- Rarely receive any direct feedback on documentation and he does not expect feedback either.

4.1.4 A4, Alpha

A4 is a developer in the company Alpha. He has over fifteen years of experience as software developer and two years in the current company, where he now feel that he has started to learn the habits and practices used within the firm. A4 does not have a same background as the typical software engineer with a university education in computer science he has instead himself learned to become a developer from an education as building engineer. Just as for the other employees in Alpha they follow an Agile philosophy where they have customised and picked what they consider to be the best part from Scrum. A4 emphasized that they are not using all the practices described but only what they find suitable for the projects.

Findings

A4 points to a couple of interesting observations of how documentation is actually utilised in the industry. Even though some of the reasons he provides seems to provide different explanations for the same problem it still provides pieces to the puzzle of why high performance documentation is difficult to achieve.

First and foremost A4 strongly emphasises that documentation is not as fun and exciting task, compared to solving problems and implementing these solutions into the code with elaborate algorithms. He points to the importance of documentation when you are new to area or in a certain module, however his scenario we feel fail to describe of when and how this documentation should actually be produced. Furthermore A4 throughout the interview base much of reasoning on the fact that the documentation existing in the firm is of high quality, but they do not have much documentation in support for maintaining the system. However, A4 thinks that if he is going to do some maintenance work, he prefers to read the code for details because he feels that you cannot trust the documentation.

A4 does not see documentation as beneficial, and therefore he does not believe that there is a problem of lack of motivation for producing technical documentation. He also points out that they are a consulting firm and the customer does not require them to produce technical documentation. Nevertheless, he admits that people is reluctant to write documentation because is boring.

When feedback was brought up as a discussion topic, A4 is fairly indifferent about feedback both regarding code and documentation. However, A4 feels that people do not have enough long term perspective for documentation. Additionally he adds that customers are more willing to for pay end user documentation, how to use the system, instead of technical documentation. This reality then contributes to the fact it is easier to focus on the short term perspective. A4 also identify that is is important that documentation is updated regularly and that out of date documentation might only create confusion and actually be of negative value.

Key Findings

- A4 recognises that the documentation is not as fun as programming and that developers tend to stay away from it.
- When you are new in a certain area, documentation can be very beneficial in order to learn about the architecture, interfaces and to understand specific functionality faster.
- When it comes to maintenance, the documentation is not trustworthy enough and he consults the code directly instead.
- Lack of feedback is not something that A4 consider to any real impact on the performance of the documentation
- A4 feels that out of date documentation, even if just slightly, is just confusing and can actually bring negative value.

4.1.5 B1, Beta

B1 is a developer in the software company Beta and has over twelve years of experience from the software industry and Beta is the company he have spent his entire career in. He has a background of a computer engineering from Chalmers and he feels that documentation is the part from his education that he did not get from school. He describes Beta as an Agile process organisation and that they are using a version of Scrum.

Findings

B1 points out many interesting facts and observations during the interview. Firstly he recognises a lack of documentation as a source of misunderstanding and an unoptimised development process. More often than not colleagues will refer just to the code when he wants to understand a certain module. A general comment for not having enough documentation is that the solution is already in the code. B1 describes that even when someone invest a lot of time investigating and understanding a certain module, it is unlikely that the same person do spend the time to actually document the knowledge they did acquire. In addition B1 describe that with documentation the readability and understandability of a very efficient and complex code can improve.

Furthermore there is also a differentiation when it comes to increasing documentation efforts, to produce more documentation or to produce documentation of higher quality. B1 recognises that while the company would indeed benefit from more documentation the quality of the documentation is a more emerging problem. Many developers including himself are unsure of what to classify as good documentation and therefore also unable to produce it at a consistent basis. The lack of guidelines about how to produce documentation is a fact that further emphasis this problem. B1 feel that documentation overall can provide much needed clarification of complicated issues and and an overall view for the interaction in a complex structure. In addition people are usually focused on the short term perspective and get their solution to work, neglecting the documentation part of the task.

Finally when feedback was brought up as a discussion topic, B1 recognised that as an area for which a lot of improvement could be made, especially on how positive feedback can be communicated along with the negative. B1 explained that it is important to have some structure or system to facilitate the constructive criticism that the developers have but are not communicating. B1 also recognises that bad feedback can be hurtful and that there might indeed be cases where not saying anything is better, if all you can provide is complaints without anything constructive. The culture within the company does not make it an immediate problem. He also feel that there feedback is never so detrimental that people might feel intimidated by the feedback to a point where they are afraid to produce documentation. However from interviewee's responses there seems to be dire need for a communication system or protocol for the feedback, to facilitate constructive criticism.

Key Findings

- B1 recognises that more documentation could be beneficial but the more emerging problem he feels is to make the documentation they already have more useful.
- B1 recognises that there is a lack of guidelines for how to produce good documentation.
- B1 feels that with a well-defined strategy for documentation the motivation would improve.
- B1 describe that with documentation the readability and understandability of a very efficient and complex code can improve.
- Developers are usually focused too much on the short term task and B1 believes that change needs to be facilitated from or with support from the firm.
- B1 recognises that bad feedback can be hurtful and sometimes is better not to give or receive feedback if is not constructive at all.

4.1.6 B2, Beta

B2 is working as a test architect for non-functional tests at Beta and has over seven years of experience in the software industry. He has a background of a computer engineer from Chalmers and he studied the Master of Network and Distributing Systems, so he did not take any pure software engineering courses. As a result of this he feels he would have liked to receive some training about documentation in Beta. At Beta they follow an Agile philosophy where they have customised and picked what they consider to be the best part from Scrum into a quite detailed and rich process. B2 describes that they emphasis a definition of done and all its properties is enforced throughout the development.

Findings

B2 feels that documentation is an important part of the development process and it is also part of the definition of done so in order for features to progress to the next state they need to have the appropriate documentation. B2 has an interesting standpoint on documentation from his role as an architect. First of all he recognises that he trust the documentation. If he want to understand a module within the system he first tries to find the answer in the documentation and secondly if his efforts was not successful he will try to find the person who wrote that document. B2 think that outdated documentation have some value because at least you understand the module at an average level and you also know which person who modified the code last. Nevertheless, at Beta other people argue that it should be a timer ticking on the document and if the timer is up the document should be deleted, since it out of date and do not add any value. He also admits that developers currently see some value in writing documentation but is like they do not feel the same responsibility as for the code.

B2 is unsure when he is asked about the reluctance from developers producing documentation. For example, in customer documentation, developers see it is important

because the customer needs it. But if it is the testing documentation they can feel a bit less motivated to produce it. The system can have very long life-cycles so it is generally very rare that the same people developing the system is also the one maintaining it and even more rare for people to remember the intention behind certain design strategies without it being documented.

Finally when feedback was brought up as a discussion topic, B2 recognised that people like feedback but only as long as they have the feeling as well as the formal permission to analyse and adapt to it. When put into a stressful situation where they already feel that they have more than enough tasks to finish they dislike it and try to discard it. Even though what they will achieve be substandard compared to what the product is expected to deliver. B3 however feel the problem is not easy to solve and that it seems to be routed in the culture that you have to deliver and keep your deadlines. Instead of realising that this is not going to be good enough, already before the deadline and take the feedback into consideration to produce a solution that is actually good enough. Instead of wasting time on the substandard implementation just to hand something for a deadline. However a change in this area would perhaps require a conscious effort from all in the chain, managers, customer and other stakeholders, where they all agree and both support and encourage that relevant feedback is incorporated into the solution even when come close to an emerging deadline.

Key Findings

- B2 identifies documentation as important and appropriate documentation is part of the definition of done.
- It is rare that the same people developing the system is also the one maintaining it making it important to document the intention behind design strategies.
- Developers do not see the same value for documentation as they do for code. Then at Beta developers do not take the same responsibility for the quality of the documentation as for the code.
- There is some value in slightly outdated documentation if it can provide an overview or general understanding of the problem.
- Feedback it is usually beneficial and also appreciated. However if people are already stretched to their maximum or deadlines prevent them for actively be able to adapt or solve the things pointed out by the feedback they tend not to like to receive it.

4.1.7 B3, Beta

B3 is a developer and scrum master in the software company Beta. She has three years of experience in the software development. She describes Beta as an Agile process organisation and that they are using a version of Scrum.

Findings

B3 has a couple of interesting observations about documentation. First she sees a lot of value having the code documented or commented even though she herself might know how the code she produced work. However as soon a team member need to understand it, problem arises and usually with maintenance even people outside your team with no insight into the context has to understand and update the code. She often consults documentation, before diving in the code because comments are much easier to read because they are written in a natural language. B3 does however think it is good that documentation is required because not everyone shares the same mentality about the importance of documentation. She acknowledges that not having the requirement would probably result in a much more unreliable coverage for the documentation. She consider a reliable coverage for the documentation an important property for a working documentation repository.

B3 does feel that the customer could generally benefit from them producing more documentation however they would not know if they produced less documentation than optimal. Moreover, the customer would never request more documentation because they do not know what she does in her daily work; they just buy the finished product. However it does feel that B3 describes an indirect lock in effect where the customer does not want and should not dictate how much documentation is produced. However their expectancy of code in the finished product restrict Beta's ability to increase the amount of documentation without taking a considerable risk themselves.

She believes that feedback is necessary for documentation because when you receive feedback on for example your comments you can realise what the others found easy or hard to understand, although for you as author it is all easily understandable. B3 also said that both code and documentation are committed through the same tool, so they should both be given the same attention in terms of feedbacks although she recognises that occasionally, especially under time pressure, code is given higher priority providing feedback for. B3 believes that when documentation is produced if a deadline is close or they have a lot of work to do, that sometimes developers do not take the appropriate time to understand the solution to provide optimised and descriptive documentation for it. That could explain the occurrences of documentation or comments she has seen that is more or less repeating the function header or code without adding any overall perspective.

Key Findings

- B3 consult the documentation/comments first when trying to understand a module as it communicates in a language that is natural to her as a developer.
- B3 does think it is good that documentation is required because not everyone share the same mentality about the importance of documentation.
- B3 does feel that the customer could generally benefit from them producing more documentation however they would not know if they produced too little documentation than is optimal.

- Since both code and documentation are committed through the same tool they should both be given the same attention in terms of feedbacks.
- B3 believes it happens that people do not take the appropriate time to understand the solution to provide thoughtful documentation for it.

4.1.8 B4, Beta

B4 is a developer in the software company Beta and has one year and five months of experience from the software industry as well as the company, as he have spent his entire career here. He explains that at Beta they work in sprints, Kanban and a version of Scrum, although they do not call it Scrum.

Findings

B4 recognises that producing good documentation is really hard and not something you can causally complete. When asked about the documentation activity requiring a different skill-set he showed some uncertainty and did find that perhaps it do to some extent. Although he found it hard to separate as it should ideally overlap with the programming skill-set. Thus it must be something possible to acquire given people practice and continuously let them produce documentation.

B4's experience is that self-explanatory code is not a sustainable and realistic long term alternative. Documentation provides an understanding on a higher abstraction level than the code. Even readable code do not and should not focus on producing understandability for this higher level of abstraction, instead documentation is a great addition without diverging the purpose of well written and readable code. Documentation is what allows for developers to efficiently gain higher level of understanding. B4 identifies that you need documentation the most when you are new to a certain area and that documentation and readability of the actual code serve two different purposes. However their outcome both aggregate towards the same goal of achieving understandability of the module investigated. Documentation is ideally first consulted to bring an overall picture and then the code can provide more detailed information about specific pieces.

He feels that there is some reluctance towards producing documentation; however in the component he is currently working people want to maintain the high quality. B4 describes that there is a common understanding that for it to happen in a long term perspective documentation needs to be produced and updated. B4 feels that in the right culture and organisation documentation becomes a social responsibility that people actually feel encouraged to undertake. However in order for this social responsibility to be sustainable for long term maintenance a common and shared ownership of the artefacts needs to exist. Developers need to care about what they produce without becoming limited and overly protective of the code they produced themselves.

B4 finds that the activity of producing documentation is not a waste of time because when someone wants to acquire the same information as you previously acquired and

do not have access to documentation you are both going to spend time on the same investigation process effectively wasting time. B4 do however acknowledges that time pressure stemming from emerging deadlines is a very real risk that indeed can affect the quality and amount of documentation that is being prioritised.

Key Findings

- B4 recognises that producing good documentation is really hard and not something you can casually complete.
- B4 identifies that you need documentation the most when you are new to a certain area.
- B4 feels that there is some reluctance towards producing documentation.
- Documentation provides an understanding on a higher abstraction level than the code.
- B4 feels that in the right culture and organisation documentation becomes a social responsibility that people actually feel encouraged to undertake.
- B4 has the opinion that producing documentation is not a waste of time, the real waste is when someone wants to acquire the same information as you without having access to any documentation.
- Time pressure stemming from emerging deadlines is a risk that can affect the quality and amount of documentation produced.

4.1.9 C1, Gamma

C1 is a developer in the company Gamma. He has around half a year of experience as a software developer. He has a background of a computer engineering from Chalmers where he remember taking one course related to documentation. At Gamma they use a little bit of mix between different Agile methods and that they are using a version of Scrum.

Findings

C1 in his role as junior developer in the company Gamma admits that junior developers are assigned more of the documentation tasks. One of the explanations he give is that due to him being new in the company they have to teach him about the processes and code base and then they realise that there is a lack of documentation and then he is tasked with producing it. He likes to document only what is necessary and keep it lightweight and has no particular interest in documentation.

C1 describes that the documentation produced is mostly of a functional nature and even though each feature is not very complex. The size of the system combined with the fact that the project have so many different nationalities, cultures and teams located in different time zones makes documentation essential. C1 does find that there exist a difference status around tasks such as documentation and coding. There is according to C1, although unformulated, clear ranks depending if you are producing code or documentation. C1 feels that documentation has a lower status in comparison to the code. He admits that some developers have the technical competence to

produce documentation but a lack of interest. C1 does believe that people are not enthusiastic about having to change their work habits that you have to sell in a new concept really well if you want people to adapt into new practises. C1 in addition also explains how the pressure to always deliver code is the inhibitor that influences their motivation for producing documentation since it takes away time from what is considered important from the customer perspective.

In order to incentive software engineers to produce good documentation he suggests Gamification as a possible solution to solve problems connected to developers not writing documentation. Furthermore it is usually the code and those solutions that are awarded by the managers and appreciated by the customer.

Finally, C1 does think that feedback is something that is heavily tied to the cultural values and attitude of the organisation. In his current workplace he cannot expect to receive written feedback and even less when it comes to positive feedback. Generally you should be happy not be yelled at, as that is not uncommon if you produce something of sub par standard.

Key Findings

- As a junior developer get assigned more of the documentation tasks primarily due to the fact that he has to learn about the system.
- C1 does find that there exist different status around tasks such as documentation and coding.
- C1 does believe that people are not enthusiastic having to change their work habits and adapt into new practises.
- C1 feels that the pressure to always deliver code is the inhibitor that influences motivation for producing documentation the most.
- Gamification could be a solution to provide more incentive to produce documentation.
- C1 does think that feedback is something that is heavily tied to the cultural values and attitude of the organisation and that he does not feel he cannot expect to receive written feedback and even less positive feedback.

4.1.10 D1, Delta

D1 is a business area manager for integration, he is a project manager for the software of Delta and he also gives support to the sales organization. He has over thirty-seven years of experience in the software industry. At Delta they have their own methodology in terms of development process, it goes all the way from how to organize work, how to name specific artefacts in a specific technical environment and everything in between.

Findings

D1 has a picture of developers from his role as a business area manager for integration and project manager that is very interesting. As they are heavy integration

focused company D1 describes that the need for documentation is great and feels that they try to produce code without having the right documentation is like trying to building a house without blueprints. D1 explains that he did consult documentation all the time and in more recent time he has been more curious of why it is so hard to make people motivated in producing the documentation. He explain that lately he has been even more interested in why is so hard to get documentation done, the dynamics around how you can make people to perform those tasks. When D1 himself realise that someone beside him will need to understand the code he usually have nothing against producing documentation for it.

He mentioned that Delta also has to point out to their customers the importance of documentation for them as they generally undervalue the existence of documentation. So while it could be convenient to go with customer view as most developers are not excited about the prospect to produce documentation. D1 feels that it is his professional responsibility for him to inform and try to make the customer understand the importance to have the documentation. Especially if their deal is somehow terminated as it not feasible to accommodate the knowledge transfer without proper documentation. When asked about if they need to or should increase their documentation effort, D1 explain that they do not need to produce more documentation but more reliable produce the documentation that is dictated by their process to be produced.

D1 ultimately thinks that problem generally comes down to a lack of governance or framework about how and when to produce the documentation. If not provided the decision would be left to individuals and will lead to a very inconsistent coverage of documentation. As the result it completely up to the individual's' motivation and idea how to best describe what they have done. When working with complex code with interdependencies in many layers D1 points that this is simply not good enough, if you want to stay on top of the process and keep it manageable.

Key Findings

- D1 is curious of why it is so hard to make people motivated in producing the documentation.
- D1 realises that is generally a challenge to motivate people to produce enough documentation.
- D1 feels that it is his professional responsibility to make the customer understand the importance for them to have the documentation.
- D1 feels they more reliable need to produce the documentation dictated by their process but not necessarily more.
- D1 thinks that problem generally comes down to a lack of governance or framework about how and when to produce the documentation.

4.1.11 E1, Epsilon

E1 is a developer in the company Epsilon. He has around five years of experience from software development where three of those years are from consulting and then

he has one year and a half of actual employ software development. He describes Epsilon as an Agile process organisation and that they are using a version of Scrum.

Findings

E1 believes that working with experienced colleagues and learning how they produce technical documentation is more efficient than education about software documentation. E1 recognises that documentation is of extra importance when it comes to maintenance compared implementing the code in development. As the maintainer usually will not have the possibility of asking other knowledgeable team members directly if it is something they do not understand. He feels that documentation out of date has very little value and sometimes just create more confusion. E1 points out he wants to use documentation to work faster, as he can read only the documentation in order to use the code module. However if the detect that behaviour of the code have changed in any way compared to what is stated in the documentation he is going to read the code altogether anyway. However he points out there is a worst case scenario, that he later realises that he has trusted something that proved out of date. If it is just a name that has changed there is really not that big of a problem and the documentation can still be used. His consideration of out of date documentation is that behaviour of the code base has changed more fundamentally.

E1 feels that he rather focus on the why of a solution when producing the documentation. However he usually also provide a high level abstraction of what the code does as well. He feels that if he can convey the general idea of the code implementation many of his fellow developers have a similar understanding of how such a solution should work and can then usually work out the details. Therefore it is essential making people share the basic rationales for the function or module.

E1 does not feel that there is a reluctance of producing documentation in Epsilon, because developers usually take great pride in their work. They want other developers to use their modules and then they understand that is essential that those modules are well documented. There is a great freedom of how you can work as long as you contribute. E1 feels that it is important for the motivation that they are free to choose how to work with documentation and how much to produce. If he was dictated of exactly what properties and how much documentation that needed to be produced he would have felt very controlled and less interested to engage in the documentation activity.

However E1 also describes that there is also almost always a pressure from managers to deliver. The managers do not care at all about the technical documentation between developers. That is instead something that is kept in control between the developers themselves. E1 describes that it is a social responsibility where developers make sure to tell anyone if they leave their work undocumented as that will waste the time for anyone wanting to use that piece of code. E1 feels that it is very much a balance that you need to find when it comes to documentation. While it is very nice and beneficial to have your code documented delaying commits too long is not optimal either. While he consider there to be an optimum balance of

much documentation you should produce, he tend to generally keep a little under that balance for his initial commit of a code module. E1 points out that while he releases code to his colleagues slightly under the balance, he awaits feedback before polishing and adding the documentation for the final version. He describe that he does so partly due to it being hard to determine exactly which parts of the code needs more detailed documentation.

Since documentation is committed together with the code the developers are supposed to receive feedback on documentation as well, however mostly the focus is on providing feedback for the code. If he does not receive feedback, he does consider that it is good enough or at least someone had the chance of requesting better quality. He considers that the code will be used somewhere else eventually so soon or later the documentation will come to use so he does not think that there is a problem if no one reads the documentation in the near proximity of time.

Key Findings

- E1 recognises that documentation is of extra importance when it comes to maintenance and not just implementing code in development.
- He feels that documentation out of date has very little value and sometimes just create more confusion.
- E1 feels that he prefers to focus on the why of a solution when producing the documentation.
- E1 does not feel that there is a reluctance of producing documentation in Epsilon.
- E1 feels that it is important for the motivation that they are free to choose how to work with documentation.
- E1 describes that there is also almost always a pressure from managers to deliver and they do not care at all about the documentation.
- E1 feels that it is very much a balance that you need to find when it comes to documentation, his initial commit tend to keep a little under that balance and use feedback to determine what he need to add.
- The developers are supposed to receive feedback on documentation since documentation is committed together with the code.

4.1.12 F1, Zeta

F1 is a developer at Zeta where her daily activities including both development of new features, maintaining and deploying already existing systems. She has about one and a half year of software development. In combination with her interest about documentation and personal engagement, she has received a substantial amount of education about documentation from her studies in software engineering. Even though she has not received any training in the company she felt that she had sufficient knowledge about documentation entering the company. F1 considers that at Zeta they are using Kanban or at least following an Agile philosophy close to a standard Kanban practice.

Findings

F1 points out many interesting facts and observations during the interview. She suggest that documentation can help to bring clarity into complex parts of the code. However sometimes she feel that the problem is that the code is too complicatedly structured and that documentation would not solve the problem. In those cases she considers that refactoring of the code is the needed solution. So while F1 recognises that it can be beneficial with documentation in the right circumstances she does also emphasise that it can bring negative value when it tries to cover up for poorly implemented code. F1 points out that one of the main and perhaps most potential benefit of documentation is the fact that it allows for inclusion of new developers. However she also emphasises that this even more call for short, correct and precise documentation because the receiver in this case already is in a situation with an overflow of information to take in and understand.

F1 does not recognise a direct correlation between more documentation and value for their customer. She however identifies that Zeta providing more documentation could help Zeta to reduce the bugs introduced into the code, thus creating additional value to the customer. F1 does not feel confident to estimate whether or not it would be cost efficient in terms of developer time invested.

F1 points out that she tries to maintain an objective perspective about the amount of documentation produced. However when confronted with a scenario, being required or asked to produce more documentation than she feel valuable, she says that she recognises that it would rarely be the case, as she would use the possibility to influence on how documentation should be produced in the first place.

F1 feels that feedback is an important component in order to maintain a good quality of the documentation produced. She personally does not consider a lack of feedback affecting her short term performance about documentation too much. Although F1 does feel that developer sometimes focus on the short term perspective and ignoring long term issues. She also recognises herself that it would be hard to distinguish if there is a drop in quality over time without relevant feedback.

Key Findings

- F1 points out that documentation can help to bring clarity into complex part of the code.
- F1 points out that one of the main and perhaps most potential benefit of documentation is the fact that it allows for inclusion of new developers.
- F1 does only recognise an indirect correlation between more documentation and value for their customer and does not feel confident to estimate whether or not it would be cost efficient to produce it.
- F1 points out that she try to maintain an objective perspective of the documentation level.
- F1 feels that feedback is an important component in order to maintain a good quality of the documentation produced and to avoid developers ignoring long

term issues.

4.1.13 F2, Zeta

F2 is mainly a developer in the company Zeta but he is also a team leader in another team. He has around fourteen years of experience from software development. He did not receive any training about technical documentation and moreover he would not have liked to, he do however regret not receiving guidelines about what kind of documentation you are expected to produce. F2 describes that at Zeta they follow an Agile philosophy, and in more detail that they are using Scrum and Kanban.

Findings

F2 thinks that the documentation activity itself is something many programmers finds acceptable, but they just do not want to write unnecessary things, he does however recognise that is hard to determine exactly what will be necessary. However F2 feels that motivation about the documentation activity goes down if he is required to produce a lot of it. When a threshold is proposed for the amount of documentation that can be required without decreasing the motivation for the documentation activity, F2 does identify that this is something he recognise. His main argument for this would be that it is not really valuable to produce documentation above the threshold and since the activity itself is not funny or exciting his motivation declines.

Additionally, F2 feels that documentation is mostly there to give a general understanding of the application and if outdated documentation can still do that he still feel it provides mostly the same value. However in cases where you rely on the documentation to integrate code, application programming interface (APIs) for example, then it is important you can trust it completely.

F2 is not of the impression that adapting to produce more documentation for their code would have any significant effect on their short term productivity. He does consider that it is mostly about attitude and habits from the developers. However it is demanding to produce good documentation and to reach above the minimum level probably some training will be required. F2 feels that people do not have enough long term perspective and mostly live in the project, they are in currently in or even just the current sprint. That is also why he tries to push people gaining experience from the entire life cycle of software engineering. To make his colleagues aware of the fact that the code produced is around for a long time and need to be updated and fixed continuously during this period.

F2 feels that receiving feedback on the documentation is beneficial and an activity worthwhile to engage in. He feels that there is much to gain from receiving feedback and that it is well worth the investment to provide feedback. F2 further emphasis that it is important to provide positive feedback in order for maintaining a motivated attitude towards document your code. From F2's experience from several companies he however feels that it is quite exceptional this is performed well .

Key Findings

- F2 recognises that documentation is important to have.
- F2 feels that motivation about the documentation activity goes down if he is required to produce a lot of it.
- He thinks that documentation activity itself is something many programmers finds acceptable.
- F2 really feel that a certain level of documentation is needed.
- Even if documentation ideally is updated he still feels that documentation is mostly there to give a general understanding.
- F2 feels that receiving feedback on the documentation is something beneficial and well worth the investment to provide.
- F2 feels that people do not have enough long term perspective in their decisions concerning documentation.

4.1.14 G1, Lambda

G1 is a developer in the company Lambda. She has around one and a half year of experience from software development. Previously she has been working as a business analyst. She has a background of a Software Engineering from Chalmers and she never attended any formal courses or seminar regarding technical documentation. She describes Lambda as an Agile process organisation and that they are using a Scrum, retrospective meetings, sprints, scrum board and code reviews.

Findings

G1 describe that at Lambda they have a very lightweight approach when it comes to documentation and generally document very little. G1 do however recognise that this might be a bit due to them being such a new company and still have not had to deal with older code base. She recognise that they might adapt more documentation as their code base grow both bigger and older. But it is also because their customers do not generally request a lot of documentation nor are they particularly interested in it either.

G1 emphasis that focus is on producing value for the customer so if there is a request of documentation from the customer she has no trouble of producing it. However if the same request comes from her colleagues and she does not agree with it, then she be less inclined to comply and feel less motivated. She points out there is no lack of work so wants to be certain that it will be useful, appreciated and feel that for her own sake she does not really need it. G1 recognises that developers individual value judgement is important to consider if you want to maintain developers' motivation.

G1 recognises that she is pretty good at writing in general and also feels comfortable in producing documentation, which counters most of the reasons that would otherwise make her unmotivated. The main reason she performs well and enough of documentation is because she can allocate enough time to write coherent texts. Nevertheless, she is not going to work with tasks that she does not feel valuable

even is she writes particular good documentation. G1 conclude that if she produces documentation she wants to be certain that it is one of the deliverable that is something that someone is going to look at and appreciate.

G1 recognises that developers are short-sighted and engage in tasks mostly for the next day as G1 consider it lays in the nature of developers. She believes that feedback is important and that documentation she receives no feedback on is a waste of time because no one uses it or finds value in it. She furthermore consider that feedback could help to provide more reflection among developers. G1 also believes if there was more and faster feedback loops around documentation, the documentation work could be more motivating to perform. From that perspective she is hopeful feedback could also help developers to think more about long term consequences when taking decisions.

Key Findings

- Lambda have a very lightweight approach when it comes to documentation and generally document very little.
- G1 does agree that it is important her value judgement about documentation match the amount she are required or requested to produce for her to stay motivated to do so.
- G1 believes if there was more and faster feedback loops would be beneficial and that no feedback on documentation means no one uses it or finds value in it.
- G1 recognises that developers are short sighted and work mostly for the next.

4.1.15 H1, Omega

H1 is a software developer at Omega where his daily activity is split between feature development and handling of error reports. H1 does have around three years of experience from software development. H1 does not have any education about software documentation partially due to his educational background is not within the software field. H1 identifies that they are using Scrum as their development process and that documentation is part of the documentation of done, and that you has to adopt it to follow the documentation guidelines.

Findings

While H1 has not received any training about software documentation he feels that the ability of documenting knowledge or facts in general is something that has been emphasised through his education and something he feel good at. However H1 recognises that some training about what would be useful to document in the company's particular context would be useful.

H1 feels that at Omega they have a decent balance between code and documentation. Instead of spending effort on producing more documentation H1 feel that improving certain properties like traceability in the documentation is what focus

should be on. The traceability, for how to quickly find the correct information is one aspect he does consider they need to improve.

H1 feels that software development jobs attract certain types of personalities and especially people that like to produce code as it is often seen as one of the core tasks for a software developer. However, most people are also find it acceptable if their work involves more tasks, such as documentation to a certain extent. However if the work shifts away too much from what many considers to be the core activity, producing code. Which also was what attracted them to the job in the first place, there might be a decrease in motivation.

H1 feels that the motivation to produce documentation depends on the experience of having or not having documentation in a later stage of the project when the code needs to be maintained. Since this experience is personal and can differ widely between developers H1 consider that there is a good idea to have a policy to enforce a common level of documentation. He continues to describe that it is important that rationales of this policy are communicated so that all developers understand its necessity. H1 also identifies with the case that he only feels motivated to set aside time for documentation as long it is consistent with his value judgement about documentation. If he is required to do more than that he will produce according to just the minimum requirements, compared to trying to do his best if he feels it align with his own value judgement. H1 states if the discrepancy is big it will eventually lead to a confrontation. He also points out that this argument is generalisable to many things in life and not only true for documentation.

Furthermore H1 also considers the coding activity to be more enjoyable than to produce documentation. H1 feels that code in particular usually feels more exciting because it comes alive in a way that documentation does not. H1 attributes that mainly to how you get immediate and objective response from the compiler and that you many times directly see in which way your modifications did improve the implementation. This is is simply not true for documentation as you might never know if the text was useful to anyone. However as he recognises that the code will be around for a considerable amount of time, he consider the documentation to be important and consistent with his values for development, thus having no problem putting aside some time to document.

H1 reports that he can find work to be valuable even if he do not receive any feedback on it as people within the company is generally good at giving feedback if there is need for improvement. So while he recognises that receiving more feedback could be beneficial he also consider it has to be balanced with the time investment of giving it.

H1 considers that there might a lock in effect around documentation but that it is highly dependent on the context. He consider it to be a good balance within Omega and that they are not locked in into a non-optimum way of working with documentation. He also points that the biggest challenges in adopting or transition into producing more documentation is not the increased time investment but rather

the changes in people's habits and that the only way that for a successful adaptation to happen is if people embraces the change.

Key Findings

- H1 identifies that there is a balance between the amount of code and documentation produced.
- H1 feels that the motivation to produce documentation depends on the experience of having or not having documentation in a later stage of the project when the code needs to be maintained.
- H1 feels that software development jobs attract certain types of personalities and especially people that like to produce code.
- H1 considers the coding activity to be more enjoyable than to produce documentation.
- H1 reports that he can find work to be valuable even if he do not receive any feedback on it.
- H1 identifies with the case that he only feels motivated to put aside time for documentation as long it is consistent with his value judgement about documentation.

4.1.16 H2, Omega

H2 is a software developer and a Scrum master working at Omega. Her daily activity is split between being a scrum master facilitating the process and producing code as a developer. She has about seventeen years of experience from software development and although her educational background in Chemistry she feels that the education about software documentation was a missing component at the time she graduated.She has not received any training about software documentation within the company but does not feel that it is something she has been missing. As a Scrum master she describes that they are using Scrum as their development process however there is also additional project specific policies for documentation that they have to follow in their development.

Findings

H2 feels that documentation is good for providing the developer with a general understanding of what is going on within the code. However she is a bit split about the value of slightly outdated documentation, while it do convey a general understanding to the developer it can also give false impression about what the code does. H2 does not feel that they have a problem of producing too little documentation rather that there is some issues with the documentation that is currently being produced.

H2 reports that at Omega they have a tool that they are using for documentation which incurs a big overhead that is hard to overcome. Not only does it affect the developers' efficiency but also their motivation to produce documentation. Her experience with the tool is that it often lead to situations where they get completely stuck or experience inefficiency. One of the ways avoiding to having to deal with

the tool is to not produce the documentation in the first place. H2 however points out that most of the informal documentation is produced without the tool and that the motivation is higher and works better for this documentation.

H2 recognises that there might a certain reluctance towards producing documentation. She feels that the most important inhibitor is that developers do not have the autonomy to influence how it is produced to maintain the perception that they are making a contribution. H2 does not feel that the documentation activity in itself is that problematic, although the activity is usually not considered as exciting as producing code, most developers think documentation is acceptable.

H2 communicates that one way to help to motivate people about documentation is to just give developer more influence on how and what to document. H2 refers to the current situation where more autonomy could let produce documentation in a way they feel efficient. Instead of having to use a tool with problematic overhead deterring developers from wanting to produce documentation altogether. H2 also feels that automatically generated documentation might be good as she then would be able to more confidently trust the documentation as you can make sure it does reflect the latest changes to the code.

Moreover, H2 close to deadline feels that it is easy to succumb to the pressure of just delivering what is asked by you short term and disregard the long term perspective. She explain that it is then up to the integrity and maturity of the team to withstand the pressure and not taking a shortcut with documentation. H2 feels that it is hard to determine if developers have enough of a long term perspective.

H2 feels that lack of feedback can have considerable effect on the motivation to produce documentation and is something that should be taken into account when work flows are designed. H2 also describes that feedback help developers to keep the documentation alive and that it feels a lot more meaningful to produce the documentation when there is an active discussion about it. Also if no feedback is provided H2 generally feels that is because no one cares or has time to use it and then it feels you are wasting time.

Key Findings

- H2 feels that documentation is good for providing the developer with a general understanding of what is going on within the code.
- H2 feels that automatically generated documentation might be good.
- One help to motivate people about documentation H2 feel is to give developer more influence on how and what to document.
- H2 feels that coming close to a deadline that it is easy to succumb to the pressure of just delivering what is asked by you short term.
- H2 feels that lack of feedback can have considerable effect on the motivation to produce documentation.
- H2 feels that feedback help developers to keep the documentation alive.
- H2 feel most developers think it is acceptable with documentation although

not as exciting as producing code.

4.1.17 H3, Omega

H3 is a software developer at Omega where his responsibilities and tasks include both developing new code and maintaining already existing code. H3 also describes that the development that his team are doing is mainly focused on developing tools for other teams. H3 has around twenty years of experience from software development. H3 describes that they are using an Agile process that would be comparable to a Kanban process, however the goal is to use Scrum as the development process and is something they are progressing towards.

Findings

H3 describes that he has no specific education about software documentation but that he has taken a course in writing online documentation. H3 further explains that he has received no training about software documentation in the company; although he feels that it would have been to limited use for him. He does however consider some guidelines on how to produce documentation in order to avoid discussions, beneficial.

H3 points out many interesting facts and observations during the interview. He considers that most developers see coding as something more exciting than documentation, but that is not the only explanation why there is such a big difference in interest. H3 describes that documentation is dominated by expressing yourself and requires not only good communication skills but also a very clear understanding of what you are doing. If you doubt your ability in these properties you will lack confidence and is mostly likely just creating stress that you then try to avoid by not engaging in the documentation activity.

H3 also points out that documentation seems like an obstruction to code development rather than a helpful and beneficial product to have. H3 mainly attributes this to the fact that time spent on producing documentation is seen to take time away from producing code instead of viewing documentation as something that could help save time for comprehension of code. H3 points out developers appreciate good documentation but are not so interested and taking initiative to produce it themselves.

H3 emphasises that an investment in documentation is a worthwhile investment to him and that he personally even consider stopping the development of features once in a while to document all code and gain a better understanding in which direction the project is heading. However he feels that is not feasible with the requirement on always delivering more features that exist all throughout the company. H3 points out that their customer probably would benefit from them producing more documentation, however their customer does not believe so. Although they try to sell this idea to the customer all the time, the customer is not listening because the quick fixes are always too important for them to be willing to focus on anything else.

H3 describes that developers make sort of notes or short pieces of documentation, however does not share with the team or commit it to the formal repositories. Although H3 considers that those notes may indeed be beneficial people do not share and H3 feels it is because people feel that is work in progress and they only want to share complete work. H3 offers the explanation that it could signal incompetence or lack of knowledge. Furthermore however H3 describes this as problematic because when people are complete with the task they are usually not interested in producing the documentation either. Instead all their attention is directed towards solving the next problem. H3 points out that when developers have to produce more documentation they usually see a decrease in their delivery of new features, so they feel less efficient. Also the documentation is less directly correlated to the delivery so they also receive less recognition, from primary their managers, but also fellow developers.

Finally feedback was brought up as a discussion topic. H3 recognises that the feedback received plays a role for his motivation as he spend more time to analyse what he writes when he know people are going to read and rely on that. The most reliable way for him to know that, is to receive feedback. H3 also identifies a change in behaviour that happens close to a deadline where people respond, if all, less well to feedback. At some point the current solution is locked in as the final version no matter what. H3 describes that is probably because developers feel they have a way to relieve the pressure that the deadline bring. In this scenario feedback will impede that and feedback is therefore usually discarded at this stage. H3 also points out that it is not only feedback but also other less prioritised tasks that fall out of favour, like for instance documentation.

H3 feels that the lack of feedback can be a problem for documentation however mostly he feels that the problem is due to the form in which feedback is delivered. He thinks that when people has to give feedback, they focus on the form and therefore they are missing the most important and useful part about the content. H3 feels that this feedback focusing on the form is not very useful and might hurt developers confidence to produce documentation. Especially since the documentation is usually produced in another than their native language and might already have doubts about their ability.

Key Findings

- H3 considers that most developers see coding as something more exciting than documentation.
- H3 points out that documentation seems like an obstruction to code development rather than a helpful and beneficial product to have.
- H3 points out developers appreciate good documentation but are not so interested and taking initiative to produce it themselves.
- H3 emphasises that an investment in documentation is a worthwhile investment.
- H3 points out that their customer probably would benefit from them producing more documentation.
- H3 points out that when developers have to produce more documentation they

usually see a decrease in their speed, so they feel less efficient.
- H3 recognises that the feedback received plays a role for his motivation.
- H3 identifies a change in behaviour that happens close to a deadline where people respond, if all, less well to feedback
- H3 feels that the lack of feedback can be a problem for documentation however mostly he feels that the problem is due to the form in which feedback is delivered.

5
Analysis

In this chapter we perform a cross-case analysis where the data is analysed and compared between different cases as well as related to the theoretical context in which they are presented. The discussion and the conclusion we draw will be based both on the findings from the individual cases and the cross-case analysis.

As Briand (2003) points out in his study where he investigates how much software documentation is enough and why it in practice is poor and incomplete: "The main reason why the questions above are so difficult to answer is that they cannot be investigated analytically and require an investigation in vivo, with actual analysts, designers and programmers. It requires the empirical investigation of human processes in realistic contexts and settings" (Briand, 2003, p. 2). We will therefore to the best of our ability try to identify and present the trends and specific exceptional observations we have made, in a structured and objective way. We will also try to relate it to existing research areas that we find relevant. As we in addition to analyse the answers from the interviewees try to verify that the analysis is consistent with theoretical context and existing literature, therefore provide triangulation of the data.

The case subjects will not be placed into our model since we have not investigated the subjects level of excitement and value for the documentation task, as this was not the aim of the study. The investigation has instead focused on finding out what concepts and entities affect and influence motivation for producing documentation and if the dimensions we hypothesised seemed reasonable.

To conclude, in this chapter the reader will have the opportunity to follow the analysis of the researchers based on the patterns and trends obtained from the analysis of the interviewees answers, as well as how it connects to existing theoretical contexts. In the conclusion chapter read about the most promising findings related to the research questions where they will be summarized into a few conclusions or suggestions.

5.1 Connecting the Excitement and Value Model to self determination theory

In the section 2.9 'Excitement and Value Model' a hypothesized model with two dimensions excitement and value, (figure 2.11), is presented. However based on the

responses of the interviewees and the literature existing around self determination theory, it is possible to extend and connect motivational theory to the hypothesized model.

We do want to emphasize that the model has been developed and considered with the autonomous software industry environment in mind. Since the exact parameters of the product a developer has to deliver is usually not clearly stated, there is a decision to be taken of exactly what to include. Additionally as there is not clearly stated problem it is usually not enough for the developer to produce just a solution. The developer also has to define what actual sub problems the original problem consists of in order to implement a certain system. Hence the developer will be in an environment with great autonomy of how to carry out the task.

This environment can be contrasted to the manufacturing business with an assembly line where the task is very well defined and the environment is consciously designed without autonomy. While motivation does not disappear for distinct requests where the result our execution is supervised there is very limited options for the individual as she can either perform the task or refuse and take the confrontation. We therefore consider that our model is most relevant in the autonomous environment where the individual has the ability to prioritize herself.

The motivations do not only vary in its amount but also in its orientation and we have chosen to focus on the orientation of motivation, trying to connect it with the model, (figure 2.11), we presented in the section 2.9 'Excitement and Value Model'. Different types of motivation, intrinsic, extrinsic and amotivation correspond to different states in our model. Intrinsic motivation corresponds to high levels of excitement and thus state 1 and 3 in our model and amotivation is generally limited to low levels of both excitement and value and thus state 4.

However extrinsic motivation can be incorporated in all of them and can range from state 1 to state 4. Even more interesting is then to connect the sub-theory of self determination theory, organismic integration theory(OIT), and its separation into different levels of internalised extrinsic motivation to the different states in our model. Focusing on how the different levels internalisation corresponds and is combined together into the different states of our model (figure 2.12).

State 1: This state corresponds well to an individual dominated by intrinsic and highly integrated extrinsic motivation. When someone enjoys or experience excitement in the task that corresponds well to what Ryan and Deci (2000) describe as intrinsic motivation (Ryan and Deci, 2000). Additionally, for the task to be in the state 1, it has to be highly appreciated and valuable by primarily the firm or your boss. Alternatively we also consider that people lacking the intrinsic motivation might be in this state, if a clear majority of their extrinsic motivation has been internalized into integrated extrinsic motivation. Individuals managing to perform this integration process usually gain this perspective due to understanding the utility of the task they engage in without a controlling external pressure. That is,

to let individuals perceive that they engage and perform the tasks on their own terms. It is often crucial for the individual integrating extrinsic motivation that the value of the tasks is well communicated throughout the integration process, and is continued to be supported if the integrated extrinsic motivation is to be maintained.

Even though the two primary types, intrinsic and extrinsic, of motivation in this state lead to similar attitude and behaviour it is important to separate between them, as the origin of enjoyment or excitement is based on two different principles. One that comes from inherent excitement and one from assimilation of values, however from an external viewpoint it is very difficult to distinguish between them. In order to achieve and maintain software engineers with intrinsic motivation or highly integrated motivation; It is necessary to have an environment which supports the individual's decision of how to work, to experience excitement or interest, in combination with an acknowledgement that they are doing something important or valuable.

State 2: This state corresponds well to an individual dominated by different states of extrinsic motivation with primarily identificated regulations or introjected regulations. The individual feels low excitement in a particular task, so the employee is no longer intrinsically motivated and usually have not integrated the tasks outcome with one's self values and standards and therefore not highly integrated extrinsically motivated either. However the individual has often either identified with the utility that the task produce or at least feel it is a task worth engage in, in order to maintain self-esteem and social acceptance. Therefore having extrinsic motivation in form of identification regulations or introjected regulation.

The individual considers that the task is something valuable and appreciated by the company, and the the task is therefore often prioritized. Although the employee is usually very neutral about the excitement as it is often suppressed in favour of the wish to be compliant in order to produce a valuable outcome. However there is a risk that the fairly unexciting activity but still prioritized activity could end up at the bottom of the priority queue, being bypassed in priority by tasks of state 3 while moving toward state 4, if a revaluation of the task is done. For example a small addition to the requirements of a new and more demanding requirement, might triggers this revaluation where the old task have its value reduced.

State 3: This state corresponds well to an individual dominated by intrinsic motivation combined with more controlling forms of extrinsic motivation with introjected regulations or external regulations. Just as in the state 1 when you enjoy doing a task, you feel excitement in the task you are performing, and therefore experience intrinsic motivation. However while intrinsic motivation might exist in the nexus between the task and the individual, the environment does not provide the needed autonomy to maintain and support the intrinsic motivation (Ryan and Deci 2000). The amount of intrinsic motivation experienced is therefore often reduced to an amount where the more controlling type of extrinsic motivation takes over.

So even if the autonomy is not always explicitly restricted the lack of value communicated implicitly will convey a feeling to the developer that they are falling behind on what people expect them to be doing, limiting their autonomy. "Furthermore, not only tangible rewards, but also threats (Deci and Cascio, 1972), deadlines (Amabile, DeJong, and Lepper, 1976), directives (Koestner, Ryan, Bernieri, and Holt, 1984), and competition pressure (Reeve and Deci, 1996) diminish intrinsic motivation" (Ryan and Deci, 2000, p. 59).

However to be expected to do things or having deadlines does not necessarily need to have a big impact on your autonomy as long as it is coupled with the endorsement of those regulations. The problems arise when you do not understand the rationale for them or they seem alien to you. The lack of autonomy does not only diminish intrinsic motivation but also affect extrinsic motivation to be perceived as more controlling. However, if state 3 is compared with state 1, the main difference is that in state 3 the task will not be seen as valuable and therefore will not be prioritized in the sprint backlog because it is poorly appreciated by your superiors. Hence, the task might be motivating in its own right but not prioritized.

State 4: In this state the individual is usually extrinsically motivated with external regulations or even amotivated. When an employee is not excited doing a task and additionally the company do not see that task valuable the task is not prioritized at all and the task is not motivating to engage in.

In a more controlled or supervised environment the choices are usually limited and the employees usually perform in the task, even if it engaged with lower drive and persistence. However in a more autonomous environment where it is expected for the developer to prioritize between tasks, it is often deprioritized to the bottom of any priority queue where they ultimately often does not have to engage in at all.

5.2 Evaluation of the different states

If an organisation is susceptible to biases their might be in their interest to control and direct those biases (Kahneman, 2011). "Given the clear significance of internalization for both personal experience and behavioral and performance outcomes, the critical applied issue concerns how to promote the autonomous regulation of extrinsically motivated behaviors" (Ryan and Deci, 2000, p. 64).

Overall we consider the state 1 the best for the individual to be in and also the company since it is where they will produce the best performance. People will engage in the task with great self determination and will show persistence in the face of challenges.

However state 2 has not to be bad and especially short term both the performance and satisfaction for the individual is good. The last part of the internalization process is also very tricky to support as many of the tasks that managers want their employees to engage in are not inherently exciting or enjoyable. It therefore becomes

a crucial strategy for managers to know how to promote more self-endorsed forms of extrinsic motivation (Ryan and Deci, 2000).

State 3 are tasks that has the potential to become well engaged and motivating tasks but the value perceived and communicated about them usually hold the individual's effort back engaging in them. However before introducing changes, the company must also analyse if the task is actually valuable to them or it might simply be a honest reflection of how valuable the task is. If the company does not feel that the task is being under invested in, this state might be perfectly acceptable. We point out however that companies needs to be careful to not have tasks lose prioritization just because being perceived as not valuable, when they indeed are.

State 4 is often complicated and unproductive and however some tasks is necessary to execute short term no matter what. This motivational orientation does not have to be worthless and in a context where the barriers of products can be expanded almost indefinitely it is important that the developers are managing the prioritization of tasks in a sustainable manner. However the motivation that we theorize is behind this prioritization might not be aligned with the strategic goals of the company and it might be in their interest to better direct people's effort into achieving long terms goals. We urge companies over time to identify those practices in order to change how they are perceived and executed as our analysis is that those tasks quite quickly might be performed significantly worse.

5.3 Evaluation of the value dimension

Most of the time being asked to do something is a sign that the task you are going to do is valuable for the person requesting it. However to be asked or requested to do something does not have to convey the feeling that your work is valuable and neither does a order to complete a task, have to communicate that the task is valuable. In an environment where there is usually an abundance of tasks to complete, developers need to consider what the most valuable is among the tasks they have been asked to complete.

This fact is supported by a quote from Ryan and Deci (2000) where they illustrate the necessity of feeling an acknowledgement from significant others: "Because extrinsically motivated behaviors are not inherently interesting and thus must initially be externally prompted, the primary reason people are likely to be willing to do the behaviors is that they are valued by significant others to whom they feel (or would like to feel) connected, whether that be a family, a peer group, or a society" (Ryan and Deci, 2000, p. 64).

Kahneman and Tversky (1979) proposed prospect theory as an explanation of how people determine what adaptations or actions are valuable to engage in. What is really important in this theory is what reference point that you have. Based on the shape of the value function (figure 2.5) investing in an outcome with slightly more gains is probably going to be seen as efficient and valuable. Although investing a

substantial amount in an even better outcome will indeed be seen as more valuable. However due to fact that the value curve give diminishing return the perceived additional value compared to extra effort is probably not seen as efficient and altogether it might not seem as a good return on the investment.

If we attempt to relate this to documentation we acknowledge that we have not investigated the heuristics for what constitutes better outcomes but if we assume that a bigger effort invested into documentation can result in a better outcome. A relatively small increase in the documentation effort might be seen as valuable. Once you has established the reference point at a higher point another small increase will again similarly be seen as valuable and to go back to previous level, most likely will be seen as a substantial decrease in value. Especially considering that individuals are more sensitive to losses, referred to a loss-aversion. This is is illustrated in the function which is more steep for losses as it proposed we are more prone to be hurt by losses than we are to experience satisfaction for gains.

The prospect theory could then explain why, in order to maintain the feeling of worth throughout a change process, you have to implement a increased level of documentation effort stepwise. As you continually has to make sure that each small increase is followed by an update to the reference point. Derived from prospect theory (Kahneman and Tversky, 1979), it is crucial that the new level of practice is established as the new reference point before you introduce further increases to the documentation effort, if you want to maintain the perception of valuable changes.

In practice it means that companies indifferent of their previous level seems to find it constructive to invest for a little better documentation outcome. Just as we have observed from the interviews companies with almost no documentation think they could benefit from introducing some lightweight documentation practice. However simultaneously companies with an already decent amount of documentation, having another reference point, also think they would benefit from a little more documentation, instead of striving towards the same lightweight practice as the company with their reference point at almost no documentation.

Furthermore we observe value judgements made at several different levels of the organisation. One at the company level but also naturally individuals will make their own value judgements. So what happens when the value judgements performed by the individual do not match the overall value judgement provided by the company. To illustrate this we cite Lethbridge et al. (2003):
"Some people will argue that SEs fail to update documentation because they're lazy. Many managers have responded to this assertion by trying to impose more discipline on software engineers—forcing them to update documents. We suggest that most SEs aren't lazy; they have discipline of a different sort. They consciously or subconsciously make value judgments and conclude that it's worthwhile to update only certain types of documentation" (Lethbridge et al., 2003, p. 38). However with the emphasis on autonomy and self determination as important for people's engagement and persistence in certain task this form of imposed discipline might be

counter productive. We theorize that if the individual's value judgement is less than the companies in combination with more imposed discipline, she will feel obliged to work on certain features that she considers wasteful or at least not valuable, decreasing her autonomy. The decrease in autonomy will diminish intrinsic motivation as well as having a negative impact on the orientation of extrinsic motivation and the dominating regulation will most likely be perceived as more controlling.

However on a positive note, with Agile practices the value judgement made by the companies have shifted towards less extensive documentation processes. A adaptation which seems to generally better match the individual developer's value judgement, given the reference points many of them have for documentation. This would then mean that individuals feels less controlled and more motivated to produce enough documentation, or at least develop a more neutral attitude towards documentation imposed on them. We experienced this tendency in our interviews and that developers do not seem to associate the documentation activity in itself with dislike and therefore do not strive to avoid it.To conclude the self determination and persistence to engage in the task, risk to be reduced if the value judgement of the individual are not aligned with the company perspective.

5.4 Evaluation of the excitement dimension

People who experience inherent excitement and enjoyment for a task does so in form of intrinsic motivation and while this a strong and efficient drive in maintaining people excitement for certain tasks. We also need to consider those who are not inherently interested or excited for a task but still feel an excitement for the completion or execution of said tasks. This experienced is classified as extrinsic motivation as instead of inherent feelings depends on outside stimuli.

While it is likely that individuals that are intrinsically motivated or extrinsically motivated with integration regulation can coexist in the same environment it is important to understand their differences especially since changes to the environment might affect the two very differently. While certain properties can promote both intrinsic motivation and integrated extrinsic motivation we feel that they respond differently to removal of those properties in the work environment. While there is a necessity for certain promoting features even for intrinsic motivation we feel that it is more stable in its nature than the well integrated extrinsic motivation as it has a foundation in the inherent enjoyment that rarely change due to outside intervention. In contrast our interpretation is that integrated extrinsic motivation is more fragile and can crumble to reductions in those critical properties, as it rely on external stimuli susceptible to outside intervention.

Furthermore if we want to understand how people rate their excitement we claim that you need to consider how people evaluate their past experiences of the task in question. As Kahneman and Redelmeier (1996) pointed we must consider two parts of selves if we want to understand how people will choose between future experiences. We have both an experiencing self and a remembering self but the experiencing self

have usually no say when it comes to decision making, between two scenarios similar to what we have already experienced. Kahneman, (2011), attributes this to the fact that we choose between memories of experiences and not actual experiences and do not think of the future in form of experiences but in form of anticipated memories (Kahneman, 2011).

We will remember an event mostly from how we experienced the peak and the end of said event. An example for the developer is when producing code, the end is usually a well perceived moment, a successful state where she managed to get the implementation to work. Whereas for documentation it varies because it will take time before knowledge whether or not it was successful is available. Since the end has such a big impact on our remembering self this might help to explain why code is perceived as more favourable.

What can be done to improve this dimension and how do the two types of different selves affect how this improvement should be organised. Ideally improving the overall situation and just improve experience might be the most beneficial, but in a context of limited resources, the question will be how to spend them most efficiently to receive the most effect.

As presented in section 2.6 'Two selves and peak end rule' we can see that the two selves might be in conflict for possible optimisation. So a relevant question that we must then ask ourselves is whether or not we want to maximise the perception for the experiencing self or the remembering self. If the latter is chosen and Kahneman (2011) imply that it is, as this is the one in charge of decisions. Perhaps the focus should be on ensuring that the documentation practice end in a more successful state.

5.5 Adaptation of the dimensions over time

We theorize that individuals in a long term perspective will unconsciously adjust the dimension of the higher level to match or align with the other dimension. This will unfortunately mean that if you have an high perception for one of the dimension this will decrease, to the level of the other dimension. This is illustrated in the figure 5.1, indicated by the red arrows. The orientation of motivation will typically move from the state 2 and state 3 state towards the state 4 state.

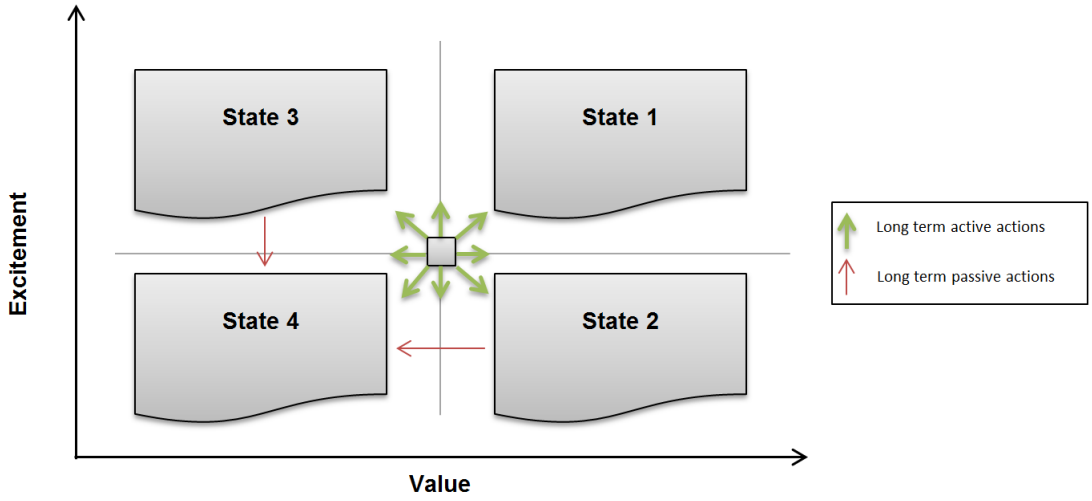


Figure 5.1: Excitement & Value Model over time

However we argue that the unconscious adjustment that influence how the dimensions are changed over time will be diminished when there is active measurement to influence the dimensions. Our observation is also that experience can to some extent mitigate the impact the unconscious adjustment have on the individual. Similarly to how the passive role, where the company is content even in the face of shortcomings will lead to this unconscious adjustment and ultimately to a worse situation. We suggest that an active role can prevent the frustration and worries perceived, regardless of experience, from having an impact on the dimensions that, we theorize determine the orientation of motivation.

While we acknowledge that actions can have both negative and positive impact we still consider an active role crucial in order to maintain motivation of beneficial orientation. This is illustrated in the figure 5.1, indicated by the green arrows. Even though the aim of the study not has been to look for different actions to maintain or improve the dimensions in accordance with our theory, we have the nonetheless identified feedback as one important action. Given the potential impact we feel that feedback can have in maintaining or improving the orientation of the motivation we find it appropriate to include a further analysis of the effects of feedback.

In order to investigate if the company was active in their role to improve the situation the job characteristics model developed by Hackman and Oldham (1976), was used in the interviews, where feedback is one of the primary job characteristics that affects the development of satisfactory and motivating work environment. We did indeed find that feedback was considered an important activity and that many appreciated to be given feedback. Many felt they was given too little feedback and especially regarding documentation who was given less feedback than activities like code production. Many recognised with consequences presented in the Job Characteristics Model and felt that feedback was a crucial part of developing a high functional work environment. Our observations are also that the underinvestment in feedback for documentation can be one of the core reason for why the improvement

process of motivational orientation has stagnated.

From the the job characteristics model also autonomy is considered as one of the primary job characteristics however most responses seems to imply that this is a perspective has come a longer way and the perception of autonomy was generally good. Most interviewees experienced that they had satisfactory autonomy of how to conduct their tasks, also for documentation, which we contribute mostly to the autonomy assigned to the teams by Agile practices.

The fact of feedback being an important supportive action is also confirmed in from other motivational research where Ryan and Deci (2000) points out out the positive impact of feedback for intrinsic motivation: "Several early studies showed that positive performance feedback enhanced intrinsic motivation" (Ryan and Deci, 2000, p. 59). As well as for the internalization of extrinsic motivation: "Thus, we theorize that supports for competence (e.g., offering optimal challenges and effectance-relevant feedback) facilitate internalization." (Ryan and Deci, 2000, p. 64).

We suggest that feedback can affect both the dimensions in our theory positively. Most straightforward is perhaps the impact that feedback has on the excitement dimensions. An alternative way to look at the excitement dimensions is to look at the memories as your aggregated experiences. While feedback does not change the actual memory they can help to consider them in a bigger or more long term perspective which could be beneficial. One of the main problem is that you produce documentation and do not know if someone will eventually read it and therefore there is a substantial risk that you are wasting time. If you however look at the bigger picture see that the a certain amount of the documentation is getting used and can consider the time saved, it can weight up for those parts that is never read.

Kahneman (2011), describes this difference in perspective when he explains that you can consider risk and chances to loss and gains with either narrow or broad framing. If you are consider it in a narrow frame you consider each case and its potential gain in isolation and if you have a broad frame you consider each case as part of whole of many more similar tasks, aggregating together to an overall gain (Kahneman, 2011). According to how we look at gains and losses we usually make more rational decisions if we use a broad frame, as we usually overestimate the detrimental effect of the single case with a narrow frame. Feedback can help you to consider documentation in a broader frame and improve how the excitement are remembered and evaluated for the individual.

However feedback can also affect the value dimension, especially when you was in doubt of how well received a certain task would be in the end. Even if the task might have been requested it can be hard to determine exactly what the result will be and therefore completed requests might turn out in a way that is no longer considered useful or valuable to the customer or company.

One of the problems are when the time to perform all of the requested tasks is not

sufficient consequently forcing the individual to choose which of those to perform. Sometimes it is implicitly left to the developer to do this prioritization themselves and most of the there is an acceptable range for which you can differ. In order to know if your prioritisation was appreciated and make a more informed decision in the future it is essential that this information is provided back to the individual in form of feedback. Otherwise it is very hard for the individual to know if the actual delivery was useful and made a contribution, something that is generally important for us, in order to continue to assign value to it in the future.

However in order for the feedback to be successful one not only need to consider what type of feedback you provide and how it is communicated but also the receiver's situation and their ability to incorporate the information. What we have generally observed from the interview study is that it depends on if you feel you have the time to adapt to the feedback. Even if you generally really like to receive feedback, if it is too late in the project and do not perceive that you can include changes within the time frame, the feedback will just add stress. As we observe from the interviews; if you are under the impression that you can relieve the pressure and then someone tells you have to change your approach and your plan becomes invalidated, it becomes easier to discard the feedback than to adapt to it. If you will not be able to change your implementation most people seem to dislike hearing that you could have done this in a better way as it is not perceived as constructive even if the content under other circumstances might have been.

However with shorter and more consistent feedback loops you increase the chances that people will feel that they have the time to adapt to changes. Additionally if you end in a scenario with time pressure where you discard the feedback, with shorter feedback loops it will be naturally be less feedback discarded each time.

5.6 Concluding analysis; Why code is engaged with better drive and persistence than documentation

We have already mentioned the properties of the remembering self and that it primary remembers the peak and the end of the experience and how that affects the excitement dimension in our theory. As code much more consistently end in a successful state because the solution is only committed when the code compile without errors and produce the desired outcome. Therefore this property has a positive impact on the code activity and to a lesser extent on the documentation activity.

Furthermore shorter and more consistent feedback loop could be an explanation for how the perception of the dimensions will evolve over time. We observe that code generally receive more consistent feedback and that the loops for feedback is generally shorter. We also consider this as an explanation for why the coding activity is appreciated better and why it is perceived in a more favourable light than documentation.

Moreover value judgements made at several different levels of the organisation is determined to have an impact on the motivational orientation of the individual. The value judgements performed by the individual regarding code production, better match the overall value judgement provided by the company, as the individual usually have a greater perception of code as valuable.

However we consider the situation to be more complex because individuals with high motivation can be both intrinsically motivated and integrated extrinsically motivated. It is deceiving because from an external viewpoint it is very difficult to distinguish between them and as all the mentioned reasons leads to better motivation, it improves both the integration of extrinsic motivation as well as enable intrinsic motivation. We interpret that the most significant difference between the coding and documentation activity is between the higher states in the excitement & value model and that code much more consistently is located at or close to state 1, documentation occasionally do but most of the time is located closer to state 2 or state 3. In order to provide an analysis that also to some extent captures the previous explanations we want to focus on the difference between intrinsic and integrated extrinsic motivation interpreted.

Ideally employees explore and find the intrinsic motivation for the task, since it allow for a much more stable high excitement level. However it will not be possible if the intrinsic motivation does not exist within the individual to begin with. It is possible that previous external regulations has prohibited the individual from experience and exercising her intrinsic motivation and that a more autonomous work environment will allow the individual to discover this intrinsic motivation. Important to note is that the word discover is used since it is of our understanding that the intrinsic motivation always existed somewhere deep within and is discovered rather than constructed. However the extrinsic integrated motivation is something that can be constructed, but not discovered, through careful use of external stimuli in the work environment. If you can manage to stimulate the extrinsic motivation in such a way that the individual adapt and assimilate the extrinsic regulations, the motivation can be perceived as self determined.

To conclude the main differences between intrinsic and integrated extrinsic motivation seems to be their response to environmental change. Motivation that has been discovered, intrinsic motivation, is more stable to environmental change since it requires the individual to forget or suppress. Motivation that has been constructed, integrated extrinsic motivation, can more easily be eliminated, as it only takes that part of the stimuli disappears.

We observe that many developers show enjoyment or excitement out of producing code. We contribute that to be primarily intrinsic motivation since the homogeneity in high levels of excitement corresponds much better to the stability against the environment that we theorize that intrinsic motivation renders. We observed high levels of excitement more occasionally for documentation and contribute that to

the integrated extrinsic motivation, as it is easier downgraded or eliminated with changes to the work environment. This difference in reaction to outside intervention, we claim could explain why the high excitement level is encountered more occasionally for documentation.

However this merely explain why an intrinsically motivating activity can continually maintain a high excitement level and why this high excitement level is stable to change in the environment. It does not explain what constituted the activity to be intrinsically motivating in the first place. Based on this analysis what is of particular interest to us is why developers so consistently seems to have discovered intrinsic motivation for just the activity of producing code but not producing documentation. Some people might have intrinsic motivation for both but from our observations they seem to be uncommon.

While we acknowledge our limitation to determine the reasons for this, the most promising explanation we have observed in the interviews is; That the software development profession attract people who have already discovered intrinsic motivation for producing code, or at least to a much greater extent than people who have discovered intrinsic motivation for producing documentation.

6
Conclusion

In the discussion section we presented the analysis of the findings in our interviews and how that does relate to the theoretical context relevant for this subject. In this section we try to summarize the most promising findings in relation to our research questions. The study tries to understand underlying factors and mechanisms influencing of how motivation affect the engagement and performance in documentation. We also try to provide some suggestions for future research based on our discoveries.

Does software engineers' motivation affect their behaviour to produce technical documentation?

Our observations and analysis suggest that the behaviour and engagement in producing technical documentation is affected by software engineers' motivation. We have proposed a two dimensional model, based on the value of the task and the excitement experienced by the individual, that determine the individual motivation. In accordance with our findings we can affirm that software engineers' motivation is affected by the excitement and value dimensions when it comes to produce technical documentation.

In order to make the differences of position in the model more distinguishable we provide 4 approximate states where state 1 is the ideal scenario and state 4 is a controlling scenario you usually want to avoid and mitigate. The motivation and perceived external regulations dominating the individual in state 2 and state 3 are usually good or satisfactory in a short-term perspective.

How is software engineers' perception of value and excitement determined, affecting the motivation for producing documentation?

We have from the theoretical context identified prospect theory, explaining how the perception of the value dimension is influenced. In our analysis of how prospect theory influence the perceived value, we have found it to be consistent with the data obtained in our study. The theory explains how people consider improvements and deteriorations in outcome and how those changes are judged in terms of being valuable. In addition we have theorized that individuals make value judgements independent of the company they are working for and that an alignment of the two judgements are important for maintaining high perception of the value dimension. The data indicates that the value judgements performed by the individual regarding code production in comparison to documentation production, better match the

overall value judgement provided by the company. This data is consistent with the obtained differences between the motivation for the two activities and we have also found support from literature for value judgement occurring among software engineers.

We have also identified the theory of the two selves as a theory that could explain the data obtained about software engineers' perception of the excitement dimension. The theory builds on the remembering self and experiencing self as two abstract but distinct entities that controls how we decide between future experiences. From the data obtained in our study we suggest it could be important to make this distinction to understand how people perceive excitement. The theory states that the remembering self is often the one having control over decisions. The primary difference between the two selves we have identified is that you have to focus on making the end well perceived due to the peak-end rule. The peak-end rule is essential to the remembering self but not important to the experiencing self.

Are there actions that can increase or decrease software engineers' motivation to produce documentation?

Our observations suggest that over time developers, if no action is being undertaken in a suboptimal environment, will unconsciously adjust the dimension of the highest level to align with the lower level. However we have indeed found that actions can increase or decrease software engineers' motivation and suggest that actions can affect both the dimensions in our excitement and value model positively and negatively. In addition active measurements or actions also heavily diminish the impact the unconscious adjustment have on the dimension of the higher level to decrease and align with dimension at the lower level. The most impactful action, significant enough, that we have discovered is feedback. We claim that receiving constructive feedback can improve the excitement of the employee. Moreover, the feedback can be helpful in order to improve the experienced value of writing technical documentation.

Is there an enabler for the activity of producing code that does not exist for the activity of producing documentation?

We have identified several potential enablers for why code production seems to be more appreciated and engaging than producing documentation. First the theory of two selves that favours code production primary because of the peak-end rule. Secondly we observe shorter and more consistent feedback loops for code produced which also favours the code producing activity. Thirdly there is also better alignment in value judgement between companies and employees for code production compared to documentation production. However individuals with high motivation can be both intrinsically motivated and integrated extrinsically motivated. It is deceiving because from an external viewpoint it is very difficult to distinguish between them and as all the mentioned enablers leads to better motivation, it improves both the integration of extrinsic motivation as well as enable intrinsic motivation.

With this in mind we have lastly identified that we consider to be the main enabler for code that does not exist for documentation. We claim that the main difference is that the code activity in high states of motivation have more intrinsic motivation compared to the documentation activity. The enabler manifests in the difference between intrinsic and integrated extrinsic motivation interpreted response to environmental change. Motivation that has been discovered, intrinsic motivation, is more stable to environmental change since it requires the individual to forget or suppress. Motivation that has been constructed, integrated extrinsic motivation, can more easily be eliminated, as it only takes that part of the stimuli disappears. Developers tendency to be intrinsically motivated by the coding activity in combination with the difference in sensitivity to environmental change; Make us suggest intrinsic motivation to be the primary enabler for high motivation towards code production that does not exist for documentation production.

6.1 Future Research

We urge future research to investigate why developers so consistently experience intrinsic motivation for producing code and not documentation. Is it that the work environment for producing code allow people to discover the intrinsic motivation so much better. Is it that the software development profession attract people who have already discovered intrinsic motivation for producing code?

References

[1] Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002) Agile Software Development Methods: Review and Analysis. Espoo, Finland: Technical Research Centre of Finland, VTT Publications ,no. 478, pp. 3-107.

[2] Abrahamsson, Abrahamsson, P. (2005) Agile software development: Introduction, current status and future. Espoo, Finland: Technical Research Centre of Finland. PowerPoint presentation Available: http://www.mit.jyu.fi/opetus/kurssit/jot/2005/kalvot/agile%20sw%20development.pdf [Accessed: 10th April, 2015].

[3] Agile Alliance (2015) [Online], Available: http://www.agilealliance.org/thealliance/the-agile-manifesto [Accessed: 21st April, 2015].

[4] Albert, J., Durepos, G. and Wiebe, E. (2009) Encyclopedia of Case Study Research. Sage Publications, Thousand Oaks, Calif 91320, vol. 2, pp. 749-758.

[5] Amabile, T. M., Dejong, W. and Lepper, M. R. (1976) Effects of externally imposed deadlines on subsequent intrinsic motivation. Journal of Personality and Social Psychology, vol. 34, pp. 92–98.

[6] Ambler, S.W. and Lines, M. (2012) Why Agile Software Development Techniques Work: Improved Feedback. Disciplined agile delivery: a practitioner's guide to agile software delivery in the enterprise. IBM Press, Upper Saddle River, N.J. 1st edition.

[7] Akin-Lagida, F. (2013) Documentation in programming, NASSCOM. [Online], Available: http://www.slideshare.net/itniketan/importance-of-documentation. [Accessed: 10th April, 2015].

[8] Barberis, N.C. (2013) Thirty Years of Prospect Theory in Economics: A Review and Assessment, vol. 27 no. 1, pp.173-196.

[9] Baxter, P and Jack, S. (2008) Qualitative Case Study Methodology: Study Design and Implementation for Novice Researchers, vol. 13 no. 4, pp. 544-559.

[10] Berger, A. (2010) Guidelines: Technical Documentation Standard for Software development. Specific Group software solutions. v1.3, pp. 6-7.

[11] Blaxter, L., Hughes, C. and Tight, M. (2010) How to research. McGraw-Hill/Open University Press, Maidenhead, England. 4th edition, pp. 135.

[12] Briand, L.C. (2003) Software documentation: how much is enough?, Seventh European Conference on Software Maintenance and Reengineering, pp. 13.

[13] Brooks, R. (1983) Towards a theory of the comprehension of computer programs, International Journal of Man-Machine Studies, vol. 18, no. 6, pp. 543-554.

[14] Bryman, A. and Bell, E. (2011) Business research methods, 3rd edition, Oxford University Press, Oxford.

[15] Capri, S. (2006) Developing Successful Software Documentation. Available: http://www.writingassist.com/pdfs/WritingAssistanceSoftwareDocumentationWP.pdf [Accesed: 14th March, 2015]
[16] Cockburn, A. (2007) Agile software development: the cooperative game, 2nd edition, Addison-Wesley, Upper Saddle River, NJ.
[17] Creswell, J. W. and Miller, D. L. (2000) Determining validity in qualitative inquiry Theory into Practice, vol. 39 no. 3, College of Education, The Ohio State University, pp.124-131.
[18] Deci, E. L. and Cascio, W. F. (1972) Changes in intrinsic motivation as a function of negative feedback and threats, vol. 25, pp. 54-67.
[19] Deci, E. L. and Vansteenkiste, M. (2004) Self-determination theory and basic need satisfaction: Understanding human development in positive psychology, vol. 27 no. 1, pp. 17–34.
[20] deCharms, R. (1968) Personal causation: The internal affective determinants of behavior. New York: Academic Press.
[21] Denzin, N. K. (1978) The research act: A theoretical introduction to sociological methods. New York: McGraw-Hill.
[22] Eisenhardt, K.M. (1989) Building Theories from Case Study Research, The Academy of Management Review, vol. 14, no. 4, pp. 532-550.
[23] Feldman, D.C. (1984) The Development and Enforcement of Group Norms, Academy of Management. The Academy of Management Review, vol. 9, no. 1, pp. 47.
[24] Forward, A. (2002) Software documentation: Building and maintaining artefacts of communication. MS thesis. Institute for Computer Science, Ottawa Carleton. Canada. ProQuest, UMI Dissertations Publishing, pp. 6-31.
[25] Forward, A. and Lethbridge, T.C. (2002) The Relevance of Software Documentation, Tools and Technologies: A Survey, Proc. ACM Symp. Documentation Eng.,ACM Press, pp. 26-33.
[26] Given, L. (2008) The Sage encyclopedia of qualitative research methods. Ringgold Inc, Portland, vol. 1, pp. 67-183.
[27] Goetz, J.P. and LeCompte, M.D. (1984) Ethnography and qualitative design in educational research. Academic Press Inc.
[28] Guest, G., Namey, E., & Mitchell, M. (2012). Collecting qualitative data: A field manual for applied research. Sage.
[29] Hackman, J. R. and Oldham, G. R. (1976) Motivation through the design of work: Test of a theory. Organizational Behavior and Human Performance, vol. 16, pp. 250-279.
[30] Hackman, J. R. and Oldham, G. R. (1980) Work redesign. Reading, MA: Addison Wesley, vol. 11 no.3, pp. 445-455.
[31] Harmon, E.G. (1997) The Case Study as a Research Method, Uses and Users of Information, LIS 391D.1, Spring 1997.
[32] Harry, B. (2011) The Importance of Feedback in Software Development. [Online] Available: http://blogs.msdn.com/b/bharry/archive/2011/06/21/the-importance-of-feedback-in-software-development.aspx.
[33] Harwell, M.R. (2011) Research design: Qualitative, quantitative, and mixed methods. Sage Publications, Thousand Oaks, Calif. 2nd edition.

[34] Hattie, J. and Timperley, H. (2007) The Power of Feedback, Review of Educational Research, vol. 77, pp. 81-112.
[35] Hiatt, J. F. (1986) Spirituality, medicine, and healing. Southern Medical Journal , vol. 79 no.6, pp. 736–743.
[36] Highsmith, J. (2004) Agile Project Management: Creating Innovative Products. Addison Wesley Longman Publishing Co., Inc. 2nd edition
[37] Kahneman, D. (2011) Thinking Fast and Slow. 1st ed, Farrar, Straus and Giroux, New York.
[38] Kahneman, D. and Tversky, A. (1979) Prospect Theory: An Analysis of Decision under Risk, Econometrica, vol. 47, no. 2, pp. 263-291.
[39] Kahneman, D. and Tversky, A. (1992) Advances in prospect theory: Cumulative representation of uncertainty, Journal of Risk and Uncertainty, vol. 5, no. 4, pp. 297-323.
[40] Kamat, V. (2012) Agile Manifesto in Higher Education, IEEE Fourth International Conference on Technology for Education, pp. 231.
[41] Koestner, R., Ryan, R. M., Bernieri, F. and Holt, K. (1984) Setting limits on children's behavior: The differential effects of controlling versus informational styles on intrinsic motivation and creativity, Journal of Personality, vol.52, pp. 233–248.
[42] Larman, C. (2004) Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional,1st edition.
[43] Larsson, A. and Persson, M. (2003) Dokumentation som undervisningsmoment i utbildningar inom systemutveckling.
[44] Lethbridge, T.C., Singer, J. and Forward, A. (2003) How software engineers use documentation: the state of the practice, vol. 20; 12, no. 6, pp. 35-39.
[45] Levy, J. (1992) An Introduction to Prospect Theory. International Society of Political Psychology. Vol. 13 no. 2, Special Issue: Prospect Theory and Political Psychology, pp. 171-186.
[46] Modigliani, P. and Chang, S. (2014) Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities.
[47] Nastasi, B. (1999) Study Notes: Qualitative Research: Sampling & Sample Size Considerations.
[48] Nawaz, A. and Malik, K.M. (2008) Software testing process in agile development. Department of Computer Science : Blekinge Institute of Technology
[49] National Center for Technology Innovation (NCTI). (2012) Case Study. [Online], Available:http://www.nationaltechcenter.org/index.php/products/at-research-matters/case-study/ [Accessed: 5th May, 2015]
[50] Patton, M.Q. (1999) Enhancing the quality and credibility of qualitative analysis, Health services research. vol. 34, no. 5, pp. 1189-1208.
[51] Patton, M.Q. (2001) Qualitative evaluation and research methods. Sage Publications,Newbury Park, Calif.
[52] Parnas, D.L. (2011) Precise Documentation: The Key to Better Software. In: Nanz, S. (ed.) The Future of Software Engineering. Berlin, Heidelberg, pp. 125-148.
[53] Redelmeier, D.A. and Kahneman, D. (1996) Patients' memories of painful med-

ical treatments: real-time and retrospective evaluations of two minimally invasive procedures, Pain, vol. 66, no. 1, pp. 3–8.

[54] Reeve, J. and Deci, E. L. (1996) Elements of the competitive situation that affect intrinsic motivation. Personality and Social Psychology Bulletin, vol. 22, no. 1, pp. 24–33.

[55] Robertson, S. (2006). Feedback in Requirements Discovery and Specification: A Quality Gateway for Testing Requirements, in Software Evolution and Feedback: Theory and Practice (eds N. H. Madhavji, J. C. Fernández-Ramil and D. E. Perry), John Wiley & Sons, Ltd, Chichester, UK.

[56] Rothbauer, P. (2008) Triangulation. The SAGE encyclopedia of qualitative research methods, SAGE Publications, Inc., Thousand Oaks, CA, pp. 892-894.

[57] Runeson, P. and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, vol. 14, no. 2, pp. 131-164.

[58] Ryan, R.M. and Connell, J.P. (1989) Perceived Locus of Causality and Internalization: Examining Reasons for Acting in Two Domains, Journal of personality and social psychology, vol. 57, no. 5, pp. 749-761.

[59] Ryan, R.M. and Deci, E.L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. Contemporary educational psychology, vol. 25, no. 1, pp. 54-67.

[60] Schwaber, K. and Sutherland, J. (2013). The Definitive Guide to Scrum: The Rules of the Game.

[61] Seaman, C.B. (2008) Qualitative Methods. London, pp. 35-62.

[62] Selic, B. (2009) Agile Documentation, Anyone?. vol. 26, no. 6, pp. 11-12.

[63] Serena Software. (2007) An introduction to Agile software development. [Online] , Available: http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf [Accessed: 2nd May 2015].

[64] Scrum Alliance (2001) [Online] Available from: https://www.scrumalliance.org [Accessed: 15th March, 2015].

[65] Showman, A. et al. (2013) Five Essential Skills for Every Undergraduate Researcher.

[66] Shinde, V. (2013) Software Testing Career Package.

[67] [67] Shuttleworth, M (2008) Case Study Research Design. [Online], Available: https://explorable.com/case-study-research-design [Accesed: 30th June, 2015].

[68] Simon, M.K. (2011) The role of the researcher.

[69] Singer, J., Sim, S.E. and Lethbridge, T.C. (2008) Software Engineering Data Collection for Field Studies. London, pp. 9-34.

[70] SME(Small-medium enterprises) (2005) The new SME definition: User guide and model declaration. Enterprise and Industry Publications

[71] Sommerville, I. (2010) Software Documentation. Lancaster University, UK.

[72] Stålhane, T. (2008) The Application of ISO 9001 to Agile Software Development. Berlin, Heidelberg, pp. 371-385.

[73] Stake, R.E. (1995) The art of case study research, Sage, Thousand Oaks, Calif; London, pp. 49-68.

[74] Stettina, C. and Heijstek, W. (2011) Necessary and neglected?: an empirical study of internal documentation in agile software development teams, ACM,

pp. 159.
[75] Trochim, W. (2006) The Research Methods Knowledge Base, 2nd edition.
[76] Welman, J.C. and Kruger, S.J. (1999) Research Methodology for the Business and Administrative Sciences. SA Journal of Industrial Psychology, vol. 26, no. 1, pp. 55.
[77] Wold, J. and McKeown, S. (2013) Business considerations in WordPress economy. Smashing Media GmbH, Freiburg, Germany.
[78] Yin, R.K. (1984) Case study research: design and methods. Sage Publications, Beverly Hills, Calif.
[79] Yin, R.K. (2003a) Applications of case study research. Sage Publications, Thousand Oaks, Calif.
[80] Yin, R.K. (2003b) Case study research: design and methods. Sage Publications, Thousand Oaks, Calif.
[81] Yin, R.K. (2004) Case study research: design and methods. Sage Publications, Thousand Oaks, Calif.
[82] Yin, R.K. (2014) Case Study Research Design and Methods. Chapter 2. Designing Case Studies: Identifying Your Case(s) and Establishing the Logic of Your Case Study. 5th edition, Sage Publications, Thousand Oaks, Calif.

A

Appendix 1 Interview Template

	Personal information
1	What is your role in [name of the Company] ?
2	How much experience of software development do you have?
3	What education of technical documentation do you have?
3.1	Have you received any training in the company related to technical documentation?
3.2	Do you feel you would have liked to receive training?
4	How much time do you spend writing documentation daily?
5	How often do they consult the available software documentation?
	Company information
6	What development process or processes do you use in the company?
6.1	You have any additional documentation policies?
7	When are you supposed to document your code? (Before/During/After)
7.1	When do you document your code?
	Personal opinion
8	Would your customer benefit from you producing more documentation?
8.1	Are you producing the optimum amount of documentation in the perspective of the person buying your products?
8.2	How complex and big is the project/product you are working on? Lines of Code, people assigned to it?

9	What benefits do you think documentation can contribute with?
9.1	Do you consider documentation helps with understanding the code and ease the maintenance work?
10	So when other developers wants to use your code, can they trust the documentation alone or do they have to read the code itself?
11	What about documentation that is out of date? Is that of any value here in [name of the Company]?
12	Do you provide documentation/comments because you are required to or because you find it beneficial?
13	What are the most important inhibitors in your opinion to why developers do not produce documentation?
13.1	What about lack of feedback, would you consider that an important inhibitor for not producing documentation?
14	Do you like to receive feedback in general?
15	Do you have feedback loops for documentation?
16	How is feedback affecting your documentation performance?
16.1	Would feedback help you produce better documentation?
17	What conclusions do you make of documentation for which you receive no feedback on?
17.1	How does this affect your motivation to produce documentation in the future?
17.2	Do you feel that “no news is good news”?
17.3	Would the notion that someone read your documentation make you more motivated to produce it in the future?
18	Is it possible to see who the author of a document is or who is responsible for updating document, so you know who you should give feedback to?

19	Do you feel that you make a value judgement about documentation?
19.1	As long the company's value judgement match yours you are motivated to work according to how they specify?
	Do you believe that developers don't see the enough value of documentation to produce good documentation?
20.1	Do you believe developers in general consider the long term benefit of documentation when deciding on the amount that should be produced?
21	Have you considered the opportunity of producing documentation and that it might be too great?
22	Do you consider that there might be a similar lock in effect around documentation as between the qwerty and dvorak keyboard layouts?
23	Are you familiar with Hackman and Oldham "Job characteristics theory" model?
23.1	If yes, how do you try to fulfill this model regarding documentation?
24	Do you feel your skills are more efficiently spent on producing code than documenting?
24.1	Do you feel there is difference status between different tasks, such as produce code and produce documentation?

Table A.1: Interview template