

DEPARTMENT OF INFORMATICS AND MEDIA, UPPSALA
UNIVERSITY

Documentation and Agile Methodology

Islam Jan and Shams Tabrez



Master in Information Systems Sciences

Supervisor: Jonas Sjöström

December 11, 2013

Abstract

Computer science in general and software engineering in specific is changing very fast. Software engineers are constantly using more innovative and more efficient ways to develop new software than in the past. This continuous evolution of software development methodologies has a great impact on both the software developed and the environment that the developers work-in. Agile software development methodologies are used to overcome many issues in the software development processes. One of the issues which still exists and needs to be addressed is the preparation of proper documentation along with the software. The work presented in this dissertation focuses on software documentation.

The work starts by a thorough literature review which focuses on different aspects of software documentation and different agile methodologies. The thesis focuses on finding out the challenges that the developers faces during their development process. Two major questions addressed in the thesis. First one is to find the motivation to document in agile environment, which is based on the hypothesis that there do exist a motivation. The second question is that how should documentation be produced such that we could avoid maximum possible potential problems. These questions are addressed with the help of different perspectives of the stockholders (i.e. developers and users) and the existing methods for documentation.

A questionnaire was developed based on the nine categories of documentation, like user documents and system documents etc.. It included different questions related to the types of documents created in software development processes, the software development stage at which the documents are created and the importance of the documents. Questions from this questionnaire are then posted on agile specific discussion forums. Where many experienced and fresh practitioners participated in the discussion. We had a detailed discussion on every component of documentation and problems were identified by the practitioners. The questionnaire was also sent to different companies practicing agile methodology. we received about 14 responses as it was detailed questionnaire with about 34 questions.

The responses of the discussion forum and survey are then analyzed and conclusions were drawn. The conclusions include that all the participants consider software documentation very important to the success of a software development project. the question of motivation is answered from the literature and opinions we received from experienced practitioners. While seven factor are identified that affect your documentation, to help solve the question of how should documentation be done.

Acknowledgment

First of all, we would like to thank Allah Almighty because without his blessings, we could not have fulfilled this difficult task.

We are thankful to our supervisor Jonas Sjöström¹ because without his continuous guidance, support and help it would have taken a long time to complete.

We would also like to thank the participants in the questionnaire, who has willingly shared their precious time during our research work.

At this stage, we are also thankful to our parents, brothers and sisters for being patience with us and offering words of encouragement throughout our thesis work.

We would also like to thank Dr. Arif Ur Rahman² for giving valuable comments on the work and providing technical assistance in writing the thesis document in Latex.

ISLAM JAN & SHAMS TABREZ

¹Jonas Sjöström, Senior Lecturer, Department of Informatics and Media, Uppsala University, Sweden.
jonas.sjostrom@im.uu.se

²Department of Informatics Engineering, Faculty of Engineering, University of Porto, Portugal.
badwanpk@fe.up.pt

*“Documentation is the castor oil of programming.
Managers think it is good for programmers and programmers hate it!”*

Gerald Weinberg

Contents

Abstract	i
Acknowledgment	iii
Abbreviations	xv
1 Background	1
1.1 Related Work	1
1.2 Research Question and Purpose of the Study	3
1.3 Dissertation Outline	5
2 Research Methodology	7
2.1 Definition of research	7
2.2 Research Methodology	8
2.2.1 Exploratory research	8
2.2.2 Empirical Research	8
2.2.3 Constructive Research	8
2.3 Qualitative Analysis	9
2.3.1 Literature Review	9
2.3.2 Empirical Study	10
2.3.3 Formulate Questionnaire	10
2.3.4 Structural Interview	11
2.3.5 Comments or Opinions through discussion forum	11
2.3.6 Identify Documentation Components	11
2.4 Design Research	12
2.5 Reliability and Validity of Research	12
2.6 Limitations	13
3 Documentation Introduction	15
3.1 Documentation and Agile	15
3.2 Process Documents	18
3.2.1 Plans, estimates and schedules	18
3.2.2 Reports, memos and electronic messages	18
3.2.3 Working papers	19
3.2.4 Standards	19
3.3 Product Documentation	19
3.3.1 User Documentation	19
3.3.2 System documentation	22

4	Agile Software Development	25
4.1	Agile Software development	25
4.2	Scrum	26
4.2.1	Roles	27
4.2.2	Meetings	28
4.2.3	Documents	28
4.3	Extreme Programming	28
4.4	Feature driven development	30
5	Literature Discussion	33
5.1	Literature Discussion	33
6	Data Analysis and Interpretation	39
6.1	Data Need	39
6.2	Data Analysis	39
6.3	Documentation Components	40
6.3.1	User Documents	40
6.3.2	Design Decision	43
6.3.3	Vision statement	44
6.3.4	Project Overview	50
6.3.5	Requirements Document	53
6.3.6	Support Documentation	62
6.3.7	System Documentation	66
6.3.8	Operation Documentation	69
6.3.9	Contract Model	73
6.4	Interpretation of Data	73
6.4.1	Type of Product	74
6.4.2	Size of the Project	75
6.4.3	The environment	75
6.4.4	Experience of the Team	76
6.4.5	Customer Requirements	76
6.4.6	Type and Level of Users	76
6.4.7	Tools Used	77
6.5	Standards	77
7	Discussion and Conclusion	79
7.1	Discussion	79
7.2	Conclusion	83
A	Questionnaire	85
A.1	User Documents	85
A.2	Design Decision	85
A.3	Vision Statement	86
A.4	Project Overview	86
A.5	Requirements Documents	87
A.6	Support Documents	87
A.7	System Documentation	87
A.8	Operations Documentations	88
A.9	Contract Model	89

CONTENTS

ix

References

91

List of Figures

2.1	Research Methodology	9
3.1	The usefulness of documentation Rüping (2005)	16
3.2	Different types of user documentation Sommerville (2001)	20
4.1	Scrum	27
5.1	How effective do you consider finding internal documentation? Stettina and Heijstek (2011)	35
5.2	How do you feel about documentation at work Stettina and Heijstek (2011)	36
5.3	How important do you consider documentation for your project? Stettina and Heijstek (2011)	36
6.1	Dsign Decision	45
6.2	Different types of user documentation	49
6.3	Identify Risks and Attack	50
6.4	Vision Statement	51
6.5	Requirements Documentation Graph	61
6.6	Factors which affect documentation	74
7.1	Operations to produce documents	84

List of Tables

5.1	Descriptive variables team results (χ) and agreement(σ^2) Stettina and Heijstek (2011)	38
-----	--	----

Abbreviations

ADT	Agile Software Development
CCC	Card Communication Confirmation
XML	Extensible Markup Language
XP	Extreme Programming
FDD	Feature Driven Development
HTML	Hyper Text Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IEC	International Electro technical Commission
ISO	International Standardization for Organization
PM	Project Management
RiPD7	Rapid Production of Documentation in 7 steps
RM	Research Methodology
UML	Unified Model Language

Chapter 1

Background

This chapter discusses about the background and agile documentation. An introduction of the documentation in agile methodology and what the practitioners and writers approach is, towards light documentation versus heavyweight documentation. It's an introduction of the problem further stretching to the definition of the problem questions..

=====

1.1 Related Work

The perception of having lesser documentation in agile is due to it is emphasis on the deliverance of the product. One of the four basic Agile principles is "‘ Working software over comprehensive documentation"’. It does not mean no documentation at all, but rather avoid unnecessary documentation for efficient production. Document in traditional way of development was meant to have a sophisticated way of development process. Which could help decrease the amount of errors and boost efficiency of the product development process.

According to Parnas Documentation is something written after the software has been developed. Most of the software developers thinks that documentation is a collection of wordy, unstructured, contain thousands of pages that nobody wanted to write and nobody trust ?. The advantages of documentation varies from one product to another similarly the consequences of lack of documentation also changes along the product. Any product’s future maintenance depends basically on the product’s documentation. Lack of documentation means problems in maintenance and having no documentation means more problems. besides maintenance during the development process better documentation helps a lot new members in the development team. Especially internal documentation helps in understanding of the tasks by all the members, every one knows who is doing what and also it has a great impact on the coordination among the team members. For example, if i know what exactly my team member is doing then its easier for me to produce what he needed from me. Similarly user documentation is also very important, in fact in some cases your perfect working product will be of no use if the right type and right amount of user documentation

is not provided. Its like removing all the colors from the color coded wires and then asking the technician to fix the problem. There has to be color then only can it be fixed. Color actually was documentation provided along the cables in an established system.

Agile do discourage unnecessary documentation but then the problem is that which document is important and which is not important. Huge and voluminous documentation could also be confusing and may not serve the readers. Sometimes project documentation are quite lengthy and poorly organized. If reader try to read it and it will take long time before find the information he/she is looking for. So the organization of information is also important.

Here the focus of the thesis is documentation in agile way of development. To find out the problems the practitioners face and how should documentation be prduced, in order to minimize or solve the potential problems. Even some of the teams are practically having no documentation. So it is very important to find the causes, why such extreme points of having unwanted documentation and having no documentation at all. Try to clarify how to avoid lengthy documentation and how to focus on concrete,concise and structured documentation. Documentation play an important role in projects of different sizes, especially large projects. Such documents usually describe user requirement documents, architecture documents, design decision, source code and management issues. Documentation mostly contributes to the projects by making necessary information available to the team members. Documentation also preserve knowledge for the team members by leaving important information for new or inexperienced members of the team or executives.

Technology has developed and it has the ability to access, store and analyze large amount of data. So it is hard to filter what we really need. Project contain a number of documents and team members looking for specific information can easily get lost. Agile documentation will be effective when it is light weight and that it does not contain unnecessary documentation. It will provide all the information relevant to readers. Such documents will be of high quality, accurate, up-to-date, highly readable, concise and well structured [Rüping \(2005\)](#).

The division of internal and external documentation is established from the very beginning of the development process. The reason for internal documentation was to make things easier for the developers. While the reason for external documentation was to make it easier for the users, who will be dealing with the product. Kent Beck suggest that well written code itself a form of documentation [Hoda et al. \(2010\)](#). Best Agile practice suggest that producing just enough documentation on right time for the right audience [Aguiar \(2009\)](#). So in agile the document should be written for the intended readers, not because that the process dictates it for writing. At the same times, documentation also suffers from the problems such as non-existent or poor quality of the document, it could be Out dated and may be the document is without definite objective ?.

Although agile methodology is a people oriented methodology while communication play an important role in software development, but it does not mean that written documentation should be neglected. [Mazni et al. \(2010\)](#). The purpose of the documentation is to capture knowledge of the software project and provide flexible and effective way of recording informal contents. A large amount of effort is always required to maintain the consistency of the documentation [Correia et al. \(2009\)](#).

1.2 Research Question and Purpose of the Study

So we are adopting two different methodologies, literature review and empirical study to answer the below give questions.

1. Q: How should documentation be produced and used in agile development?

The first question is to discuss that how documentation in agile methodology should be produced. We will try to answer this question from the literature with the help of the identification of all possible problems practitioners facees during the development process. To find out that, first we need to discuss why documentation is done and why is it important to document different components of documentation. First a detailed discussion of the documentation is done, and then different methodologies that claim to be agile are discussed. To illustrate what the literature says about documentation and how the agile methodologies work according to the literature. The first question would require what really the related professionals think about documentation in agile environment. Which components they are implementing or they plan to implement in their work. This question will be more focusing on the components or different aspects of documentation. We intend to use empirical strategy here, we will conduct a survey asking the professionals that which components they think are important and implementing at work. we will also get opinions of leading practitioners to identify all potential problems.

There are no fixed documentation parts in any methodology, the documents change from one project to another according to the needs of the users and the instruction of the customer. The development environment also has impacts on the documentation. So we cannot ask about defined specific documents, as they keep change from one project to another. We can also find out which class of information is considered more important and which one is considered to be relatively less important. So the first question will be discussed and try to be answered by discussing theimportance of documentation and the working of agile methodologies. Then the different aspectsof development that the documentation depends on.

This document will not only refresh the need why developers wanted to have everything documented, but also to discuss extensively the role of agile methodology and how it discourages extensive unnecessary documentation. Now our task is to present the concept of documentation in agile methodology from the literature and then conduct the survey. Which would aim at finding out that which components or parts of documentation are implemented in practical nowadays.

2. Q: What is the rationale (motivation) to produce and use documentation in agile development?

The second question states that what is the rationale that is the motivation to produce documentation in agile methodology. The reasons for documenting a process of development and provide supporting documents are the same for Agile as in any other development methodology.

As we know that documentation has two classes internal documentation and external documentation. The importance of both these type of documentation has its due importance in the development process and in agile methodology as well. The purpose of internal documentation originally was to make the development process more transparent and to have better coordination among the team members during the development process. We need to find if this is the rationale that we still need in agile way of development. However Agile has its own structural way in the form of meetings and face to face communication that actually reduces the traditional internal documentation to some extent. It also helps to have better coordination and understanding among the team members. Similarly external documentation is to help users who will be dealing with the product, and the amount of external documentation depends largely on the users and their requirements. External documentation is mostly based on the demands of the users so its a deliverance is crucial whether it is Agile methodology or other traditional methodology.

There is general perception of documentation being less important, and the best way to negate this perception is to bring up the actual motivation behind documentation in Agile development. In traditional methodology software developers/technical writers always go for large amount of documentation while Agile motivates for less and concise documentation.

The question of what is the rationale or motivation behind the production or usage of documentation in Agile methodology is based on the assumption that there do exist a motivation to document in Agile way of working. The reason that we are convinced of the fact that documentation has its importance in Agile community, though it may vary from people to people but its never been denied of. But even if you donot agree with that argument, the question raised above would not only answer the question of ‘what is the motivation to produce documentation?’ but also automatically answers the existence of motivation for documentation among the practitioners.

When we studied literature related to the subject of documentation in Agile Methodology. None of the Agile researchers or practitioners had an opinion of documentation as something less important. In fact each one of them had a different argument on how documentation is important, but of course opinion differs when it comes to how detailed documentations should be or which components are more important than the others. Over all there existed an encouraging motivation which is quite opposite to the general perception of documentation less important in Agile.

We felt the need to bring up the question of rationale/motivation to highlight the existing importance of documentation among agile practitioners. We wanted to have an analytical discussion on the subject before answering the question. To further expand it for the sake of better understanding

1.3 Dissertation Outline

The dissertation is divided into seven chapters. Chapter 1 gives the background and discusses the research questions and how we plan to answer those questions. Chapter 2 discusses the methodologies we used and discusses how the respective methodologies will work to help answer the questions. Chapter 3 gives an introduction of documentation, Its different components and discusses it. Chapter 4 introduces different Agile methodologies. Chapter 5 discusses literature in distinctive perception from different personnel with extensive experience. Chapter 6 is about the analysis and the interpretation of the data we received in response to the survey. Discussion and conclusion are given in Chapter 7.

Chapter 2

Research Methodology

In this chapter, we have presented that which methodology we have used to gain or add knowledge. How the thesis work looks like and how the conclusion came through using different approaches. The chapter describes the research process and gives a structural look of it.

=====

2.1 Definition of research

Research is a scientific and systematic way to get information about a specific topic. The meaning of research in Advance learner dictionary is a careful investigation or inquiry, especially though for new facts in any branch of knowledge. The main purpose and the aim of the research is to discover answers for the questions raised. Different types of research have been mentioned throughout the literature. For example, descriptive vs. analytical, applied vs. fundamental, quantitative vs. qualitative and conceptual vs. empirical, etc. [Kothari \(2009\)](#) .

Now our main methodology that we used in this document is Descriptive qualitative, in descriptive research one should describe the state of affair as it exists in present situation. In this method the researcher have no control over the variables but he/she can just describe what has happened and what is happening. SERVE¹ center at University of Carolina defines Descriptive qualitative research as “detailed descriptions of specific situation(s) using document review, observations or interviews”.

Descriptive qualitative is founded in existing knowledge, thoughtful linkages to the work of others in the respective field and experience of the research-groups. The various qualitative approaches focus on various phenomena and thus produce different results. Both description and interpretation are legitimate, but they are tied to different conditions and interests” [Neergaard et al. \(2009\)](#).

¹www.serve.org

Now we have also conducted a short survey, short in-terms of target population. Please note that this survey is not based on systematic authentic sampling to represent the whole target population. This survey is just meant to support our arguments and to help understand answers for the questions. This survey should not be used or referred to in any kind of further work.

2.2 Research Methodology

The goal of every research process is to produce new knowledge or add new knowledge to the knowledge archives of global community. Addition to the knowledge could be in any of the following three forms. Our research work adopts two of the following three knowledge gaining methods.

2.2.1 Exploratory research

Exploratory research structures and identifies new problems. Exploratory research often concludes that the perceived problem actually does not exist, and is not typically generalizable to the population at large. Exploratory research according to the literature often relies on your secondary data. Which means the reviewing of the literature or already present data, or it could be qualitative approaches such as informal discussion or opinions taken from the concerned stakeholders [Saunders et al. \(2011\)](#). Which in our case would be the practitioners of Agile Methodology and their experience with the customers. We have a large portion of our study in form of exploratory research, in which we posted questions on the discussion forum and received the comments or opinions of experienced practitioners of Agile methodology. Besides that we have also a large amount of data from the literature to answers the questions we raised at the beginning, such that those commonly perceived problems actually doesn't exist and the part that does exist has a proposed solution.

2.2.2 Empirical Research

Empirical research is the second method of gaining or adding knowledge, which could be based on both direct and indirect observations or experiences. The Empirical evidence (recorded observations or experiences) can be analyzed both quantitatively or qualitatively. Through quantifying the evidence or making sense of it in a qualitative form, you can answer empirical questions. Which should be clearly defined and possible to answer with the help of those findings. The research design varies depending upon the research questions raised and the field of study [Goodwin \(2009\)](#). Our empirical study is of qualitative nature, as we have analyzed the results of our empirical study in a qualitative manner. Such that it supports our conclusions to help find answers to the two questions (problems) we raised in Chapter 1.

2.2.3 Constructive Research

It is another common research method, which demands certain validation that doesn't have to be that much empirically based as in other methods, nor the conclusions have to be so much

objectively argued and defined. It's another knowledge addition method which we are mentioning for the sake of understanding but is not part of the thesis. It involves evaluating analytically the 'construct' that's been developed against some predefined criteria or tests or prototype [Saunders et al. \(2011\)](#).

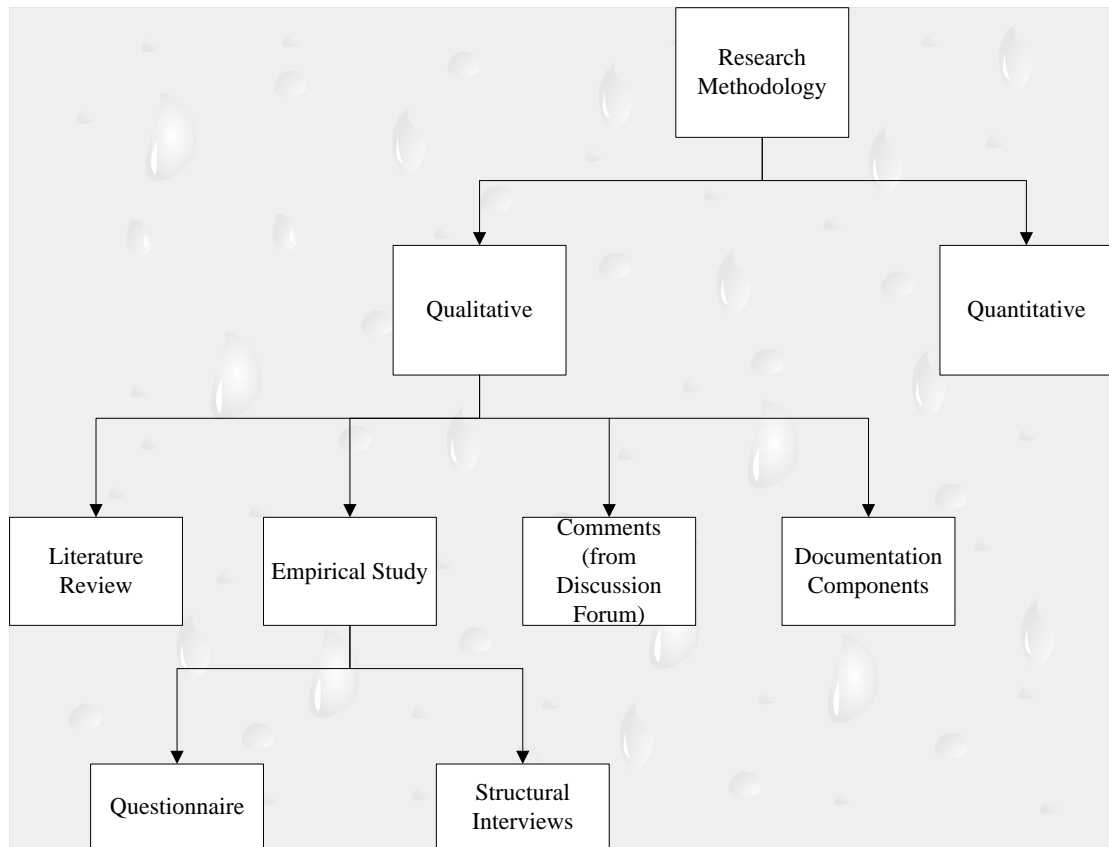


Figure 2.1: Research Methodology

2.3 Qualitative Analysis

The study is based on qualitative analysis and interpretation. The research process consist set of actions which have been performed throughout the research process. Above figure ?? shows those research processes but the sequencing is not necessary to be carried out in the specific order.

2.3.1 Literature Review

This is one of the two methodologies that we adopted as base for conducting the research study. The third chapter that discusses documentation in detail, then the fourth chapter discussing agile(the agile manifesto) and different popular methodologies that claim to be agile helped identify the 9 categories that cover all possible Agile documentation components. Finally, the fifth chapter is wholly based on literature study quoting comments distinct experienced agile practitioners and

researchers. A review of a recently done research study is also presented, which is very much related to our work. We studied enough before we started working on the thesis. Understanding of the problem domain can only be achieved through previous studies or previous work done by other researchers. Furthermore, first we had to select the problem area, and then we identified and started understanding the problem. You need to have a comprehensive study of the subject area so that you are able to look at the problem from different perspectives. Because a problem not fully understood can never be solved. After that when you have a comprehensive knowledge of the area, and a complete understanding of the problem domain, then you need to find out if there is any previous attempt to find a solution to solve the problem which requires further more study. If the answer is yes that means the problem no more remains a problem. However, if the attempt was a failed attempt, then study the failed attempt and move on with finding your own solutions. And all this could only be possible with the help of previous literature study. To identify the problem domain, to find out that the problem still remains a problem, to search that if there are any similar type of problems and attempted solutions for those problems.

2.3.2 Empirical Study

We also did an empirical study along with reviewing literature. We did a survey and sent a questionnaire to the leading software development companies in Sweden, who were using Agile as software development Methodology. We received answers from only 12 respondents which were not as much as we expected. However, as the questionnaire was a very detailed questionnaire with about 34 questions, so we used the data for analysis. Since it was hard to make statistical generalizations based on few respondents, we changed strategy and decided to make a qualitative analysis based on a discussion forum where experts discuss and give their opinions on different subjects in Agile development methodology.

2.3.3 Formulate Questionnaire

Once we have all the components of Agile documentation, then we generated a questionnaire based on those nine elements. In the beginning, we formulated a questionnaire which consisted about 34 questions, four questions for almost every component. All these questions are of different orientation. Some questions are easy to handle and you just have to answer in yes or no. Others are multiple-choice questions where you have to choose from the given choices. There are also questions where you have to check the check boxes to answer multiple answers. However, some questions were of the nature that you can't answer in yes or no. Neither is there any choice given to choose from. Those questions you have to answer in the form of description, and you have to express your thoughts. The respondents were not sufficient so then we posted questions on the discussion forum. Only thing was that on a discussion forum, you can just post one complete question about each component, and then the discussion gives you all the answers you need. We didn't have to post all the question about every component. The questionnaire we formulated is given below in the Appendix [A](#).

2.3.4 Structural Interview

Once we had the documentation in different classified categories, we then decided to ask from the developers about these parts. To find out that which parts are actually practiced in the market and which are not, and also which one they think to be important and which one is not that much important. Besides that we were also anxious to find out what really is the understanding of all these documentation components in the market, what they use specifically to implement it. So to find answers to all those questions, we divided those nine parts in a 34 questions. Some questions are dependent on the other questions so the questions has to be in structure. for example, the very first condition to fill the questionnaire is that you have to be an agile practitioner, if you are not an agile practitioner then the rest of the questions are of no use to you. Now different means has been adopted to collect answers for those questions. Although we preferred to have a in person interview with the interviewee but due to certain limitations, we have been able to get few of those interviews. We have an extensive discussion on each of the questions we raised, in the form of comments from experienced practitioners and researchers. All the means that we used to collect data are as follows.

- Face to face Interviews.
- Skype Interviews.
- Interviews through Phone.
- Questionnaire in digital form.
- Comments (Discussion forum)

2.3.5 Comments or Opinions through discussion forum

We posted questions from the defined questionnaire regarding the component of documents we defined, and collected all related comments and opinions of the practitioners and researchers. Some of these researchers are considered to be the pioneers of Agile Methodology in the Agile practicing community. it is because of their experience and the work they have done. We not only received answers to the questions we posted, but we had an extensive discussion and exchange of arguments among these practitioners and researchers. However, we have been able to include only some of the related comments, but the conclusion of this whole discussion process is the same as presented in the thesis. We have also presented some opinions from the literature which are given throughout the thesis along with references.

2.3.6 Identify Documentation Components

After an extensive study, we identified nine components of documentation. Which we believe covers every single document that a developer can think of. Now just to clarify that we are not suggesting that all these nine components should be practiced during the development using Agile

Methodology. We identified these nine categories so that we could find out what the practitioners think about each of those components. These components are presented by Scot Ambler which we find to be the best categorization. The nine components are.

1. User Documents
2. Design Decisions
3. Vision Statement
4. Project Overview
5. Requirements Documents
6. Support Documentation
7. System Documentation
8. Operations Documentation
9. Contract Model

2.4 Design Research

It has been mentioned now a number of times, but we will repeat it here again a survey has been done with the help of the questionnaire. The results received with the help of that questionnaire has been given along with each component discussed in the thesis. Besides that survey which we didn't find to be enough to generalize to the target population, we started posting the questions on a discussion forum, with the purpose of generating discussions directly related to our research questions. We received a very positive response and a lot of answers from very experienced practitioners and researchers. Then we did a structured analysis of the forum content and categorized the discussion, leading to a model of factors affecting documentation. This partially answers our questions, but the two questions raised at the beginning of the thesis are both answered with the help of both the empirical study as well as the previous literature study.

2.5 Reliability and Validity of Research

There is no clear defined set of documentation to be produced in Agile way of working. We have selected nine elements that we believe best covers all the possible documents. Now validity of the work done from the literature can be approved from the answers to the raised questions. While the reliability of the literature work covered in the selected comments from that literature can be approved from the references given in the document.

In the thesis, we have to look from qualitative perspectives, and the reliability and validity related to the qualitative concept. However, [Patton \(1990\)](#) states that both reliability and validity

are two factors which any researcher doing qualitative research should consider while designing the study, then analyzing the results and then judging the quality of the study based on that. Healy and Perry (2000) said that the quality of a study in each paradigm should be judged by its own paradigm's terms. Then terms of reliability and validity both are essential criterion for quality in the quantitative paradigms, while in qualitative paradigms the terms Credibility, Neutrality or Conform-ability, Consistency or Dependability and Applicability or Transferability are to be the essential criteria for quality [Golafshani \(2003\)](#).

First four chapters are all about the quality discussion and presents literature, we have tried to put everything in an order so that it is easy to understand. Furthermore, we have tried to include the essential information that we thought are important for solving the queries in the reader's mind. The empirical part of the thesis is only for analytical purpose. Collection and mentioning of empirical data was important, so that the credibility of being a reliable and valid source of information is clarified to the viewers. The detailed discussion about documentation and Agile Methodology may help clarify concepts about the background of the problem and the problem itself.

2.6 Limitations

Before we discuss the limitations first we would like to discuss that the demarcation of the study should be clear as it is all about documentation in agile environment. The detailed discussion in chapter 2 and 3 is for the understanding of the readers. Because we wanted to provide all necessary information along with the thesis before we discuss the problem area. WHY and HOW are the two basic questions related to documentation in agile methodology, That has to be answered at the conclusion of this study. The thesis is not about traditional documentation or documentation in other traditional methodologies. Neither is this study about the working of Agile methodology. Both these topics have been discussed and presented to the readers to have background knowledge about the area. First we conducted a survey with the help of a questionnaire, the answers taken were from professionals from different companies in different regions. However, this was a compulsion that the person answering must be working in a group or company, and currently implementing projects in agile environment. The study is based on a specific sample of the market but a diverse one. Second thing is, that although we preferred to take interviews, but we have not been able to take a lot of interviews, so the survey is based on answers through different means. That includes in person interview, through Skype, phone, and by a questionnaire in a digital form. The data received is used only for analytical purpose.

Chapter 3

Documentation Introduction

This chapter gives an introduction of the documentation, and some literature background related to documentation. It discusses different types of documentation (Product and Process). Furthermore, this chapter also highlights the schedules, Budget and standards carry out during the project to build documentation.

=====

3.1 Documentation and Agile

The specific target of this thesis is Agile methodology for software development and necessary documentation achieved in any Agile project. However, before going specifically to Agile documentation, we would like to discuss what are the basic reasons for documenting a software-development process. How should we look or approach the process of documentation and different perspectives of its stakeholders, and what do we achieve (documents at hand) by documenting a project. Alistair Cockburn recommends that documentation be 'light but sufficient'. He introduces the Crystal family of methodologies, which is targeted at projects of different size and criticality. The Crystal methodologies require documentation to be created, but the individual project decides what that documentation consisted of [Cockburn \(2006\)](#). Scott Ambler's book on Agile Modeling includes a chapter entirely devoted to documentation. He compares the Agile approach to documentation with 'traveling light' to create just enough models and just enough documentation. He also asks the right question : What would you will prefer to have a 2000 pages system document that is likely to have a significant number of errors in it, or a 20 pages of a high level overview? [Ambler \(2002a\)](#). Documentation always raise questions like that how to create, maintain and distribute documents among team members and to avoid from unnecessary documentation and unjustifiable cost [Sauer \(2003\)](#). Jim Highsmith in his book 'Agile Software Development Ecosystems warn us not to produce documentation for documentation's sake, but calls for a balance. Documentation in moderation, adds communication, enhance knowledge transfer, preserve historical information, and fulfill governmental and legal requirements [Highsmith \(2002\)](#). Andreas Ruping

also says in his book 'Agile Documentation' that light but sufficient approach is favorable for two reasons. First, such an approach prevents the project team from expending unnecessarily large effort on documentation. Second, light but sufficient documentation is more accessible and therefore, more useful for a team than voluminous documentation [Rüping \(2005\)](#). Figure 7.1 shows the relationship between the amount of documentation and its usefulness. After some particular time, the usefulness of documentation decreases as the amount of documentation increases. Then it is very hard to find relevant information as the amount of information increases. James Shore refers

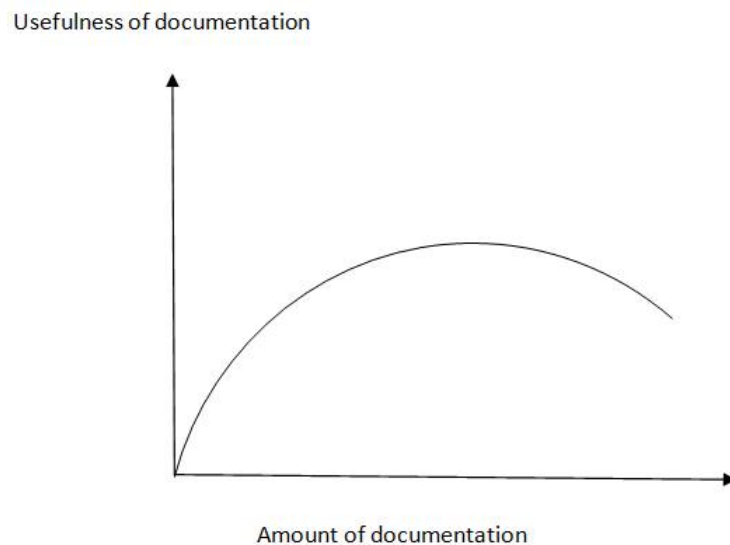


Figure 3.1: The usefulness of documentation [Rüping \(2005\)](#)

to his book [Shore et al. \(2008\)](#) and depicts the documentation in Agile environment as 'projects use three main types of documents: work in progress, product and hand off'. Now he divides the whole documentation into three categories and then defines or describes each of those categories as follows. "Work-in-progress documents communicate, and other forms of communication may replace them. High-bandwidth communication replaces some work-in-progress documents. Test-driven development creates executable low-level design specifications. Customer tests describe high-level behavior, and a ubiquitous language further clarifies intent. Product documents have business value—schedule them with stories. Hand off documents are best and most accurate at the end of the project. Set aside time after delivery to create them, and consider conducting an incremental hand off using pair programming and collective ownership" [Shore et al. \(2008\)](#). While searching the phrase like "documenting software", so hundreds of links will appear leading to different definitions or descriptions of this phrase. For example, Wikipedia describes software documentation as "written text that accompanies the computer software, it either explains how to operate or how to use it". Technically, this definition is right, but actually it is much more than just a couple of manuals provided with any software. During search one will find categorized structures on different locations given, which actually describes what how to divide the whole set of documents achieved during the process into different categories. Moreover, the nature of any particular

element of the documentation would help to find its place in the categories given. However, there is a number of such categorization of the elements of S/W development documentation. Some prominent categories are requirement's documentation, architecture/design documentation, technical and user documentation, etc. From the reader's perspective, this categorization and structural presentation of the elements of the documentation are very good. It simplifies things to the reader to understand easily. For example, the manuals along with the software are for the users/operators of the software, to help them understand the operation. Similarly different design and architecture documents are for the technical personnel, who are responsible for the development and maintenance of the software once it's ready to function. Moreover, sometime other stakeholders also need to know about the design and structure. Software producing companies are always very keen to keep the record of these types of documents, which actually helps them develop similar or more sophisticated software. Similarly, requirement documents show what the customer wanted to have and what the software should do and should look like. The above whole discussion was about the reader's perspective. Developer's perspective is very different, and by saying 'developers', we mean the team responsible for the successful implementation of the project. To achieve the bigger goal, they define some set of requirements that should be met for a successful project. Now to achieve each of these requirements, they select different elements of documentation, or you can say that during the process of achieving these requirements different elements of documentation are produced. Which are then categorized according to the readers. We would like to recommend the way Ian Sommerville described it in his book [Sommerville \(2001\)](#). He actually first presented a set of requirements that are supposed to be met by the whole set of documentation. We understand the success of your project would be to meet those requirements which are established, to achieve that one can do it on his own way. These requirements are actually the same for all types of development methodology. However, parts or elements of documentation may vary from method to method. That is because there is a possibility of achieving the same requirements through different approaches. So the focus should be on "How to achieve the requirements established". Following are the necessary requirements that must be met with the help of documenting any software-development process. We consider these to be an only way of achieving an all-time successful project.

- Work Transparency and Interpersonal/Intra Team communication
- Information repository for future maintenance
- Info's generation for time and budget planning
- Support documents for the Management and users

There are different types of documents that represent different purposes and are generated in different phases or practices through the development process. Sommerville divides it into process documentation and product documentation, which represents that all the documents produced during the development process are termed as process documentation. The documents that actually

describes the operation of the product or if the document covers only the description of the product then its product document [Sommerville \(2001\)](#).

Technically, documentation starts when you actually prepare a proposal for acquiring a tender. A detailed requirement engineering process is implemented, and a requirement document is produced. This document is a highly important piece of document that actually leads the whole development process. It defines and describes the actual features of the product and all the behaviors of the product. When the development process begins, all sorts of documents are developed during the process. As we have mentioned above that it's not a compulsion that one must produce a particular set of documents. Only the documents that you require to achieve the objectives set in the requirement document should be produced. Some procedural documents that actually help successful implementation of the process of development. If we divide and present some standard documents that should be produced in the development of any software product. This categorization is basically introduced by [Sommerville \(2001\)](#).

3.2 Process Documents

The whole development process should be organized and well planned. The division of work tasks, the assignment of time slots to those individual tasks and also assigning of the required resources to it. Each of these tasks to be achieved needs an extensive planning and paperwork before the project starts. In a process, documents are generated either before the project starts or during the process of implementation. Following are some of the types of documents generated during the development process.

3.2.1 Plans, estimates and schedules

These are the documents which usually are done or started before starting the overall project. Major plans of what product is to be generated and how and then estimating the budget and resources required, Then developing a proper time schedule. Now the more extensive the planning and the easier the implementation would be. Furthermore, it clarifies the ambiguities and provides a clear guideline, and the project becomes much more predictable [Taulavuori et al. \(2004\)](#).

3.2.2 Reports, memos and electronic messages

Every single report generated during the development process is also part of the process documentation. These reports could be either daily or weekly or monthly, all types of reports, even if it was a verbal communication. Everything should be part of the record and visible in the development process. Even informal communication and emails are included.

3.2.3 Working papers

Now these documents consist of all the technical work done, whether some technical code generated, a rough sketch of the idea, the engineer had or an email about some technical issue. Mostly, the rationale for the designs is generated in these working papers.

3.2.4 Standards

Standards actually identify the scale of the project. What standards are to be followed during the development process. These standards could be some organizational indigenous standards, followed within the organizational projects. It could also be some national standards (differs from country to country and culture), or could also be internationally established accepted and followed standards.

This is a fact that most of the documents generated during the process of development are of a very short age and are specific to that particular moment or phase. Most of them outdated and are not of that much importance later-on. Still it's kept as a record and is a vital part of the development process, and could be useful in implementing similar tasks or maintenance. However, the time and resource schedule is something that keeps changes with the progress, almost weekly and fortnightly. So the final schedule has to be produced at the end of the project. This final schedule should indicate both the original schedule at the start, and the actual time spent on achieving each task.

3.3 Product Documentation

Product documentation covers all the documents that actually describe the generated product (software) or the operation of the product. These documents can be developed during the development of the product, but mostly done in the final phase of the software-development process. The reason is that the purpose of these documents is to describe different features from the users and developer's perspective. And these descriptions can be made when you have the final product at hand. On the other hand, describe each of the features on its generation. Now each of all the documents which are considered to be part of product documentation has to be either part of User documentation or system documentation. These are two major sub branches of product documentation, which includes mainly documents related to describing the product or operation of the product. Now both these headings could have a number of different documents part of it. Some of them are discussed below.

3.3.1 User Documentation

Now we know that every single product produced has different types of users. No matter how much simple to operate your product is, you need to provide sufficient guidance material. There could be sometimes a number of different level users of a product, but we will present just two prospective users. One is the End-user's perspective and the other is the Administrator's.

1. **End users:** End users are actually the final beneficiaries who actually use the software to assist them in achieving the day-to-day objectives. That could be a manual with any machine or counting software for an accountant. Their need is to perform the operation the right way and get the job done for him. This is one of the most important documents that actually leads the users how to get started. The reason why end user documents are so important is because that you never know about the end user. The level of intellect differs from one user to another, since we are not able to help get every user started with the software. There must be a document provided that actually describes the major functions and operations of the product, and helps the user perform those operations.

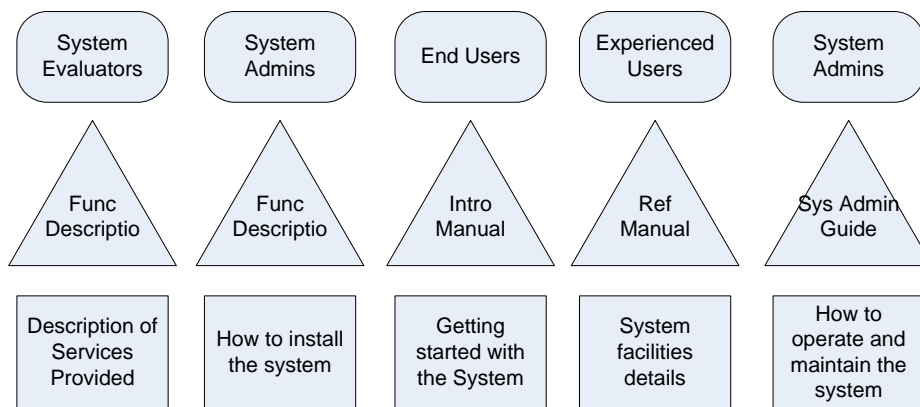


Figure 3.2: Different types of user documentation [Sommerville \(2001\)](#)

2. **System Administrators:** System administrators are responsible for the management of the software. That actually is used by the end users. System administrators normally are not considered as the genuine users (end users) of application and other software installed on the system. Their job is mainly to make sure the availability of the application and to update and maintain the product. The documents required for the system administrators are completely different from the of normal user level documents. They do not need assistance with how to operate the software, but they need some sort of documents that actually help to operate the process of installation, and also other many related things that would help him in maintenance and keeping the product updated.

To deal with the requirements of different level of users, you need to have a document for each of those classes of users. So that it could help them implement their respective operations. So here the traditional five documents we will discuss. Each of these documents is meant for a distinct class of users with a different level of expertise. These documents may vary in number or names with different systems, but the materials are all in there. In some cases, two or more documents are merged into a single document. Even sometimes you may see all these documents or the material of these documents in a single-user document. That will cover from installation to the description of functionalities and operations, etc., so each of these users can get assistance from such document. Following is the standard known user documents.

- (a) **Functional description:** This is not only the most important document, but also it's the most important phase of the development process. Most part of this document is produced in consultation with the user or owner. As the name of the document shows that this document is meant to describe the functionalities of the product. The actual reason of the existence of this product is given in this document. So the prime objective is to meet the needs of the user, and only the user knows what he/she needs. So the functions must be defined by the user or at least with consultation with the user. All extensive initial requirement work is part of this document. Besides a general description of the functionalities of the system of new users. This document is meant for the end users.
- (b) **System installation document:** This document is required to provide some guidance throughout the installation process of the system. The installation and then the configuration of the system is a complete set of operations that must be performed before you have a system that is usable at hand. This document is required by the system administrator. As most often the administrator is responsible for the installation, maintenance and configuration of the system and the applications of the system. I personally don't agree to the idea that the automated installation of different products now has decreased the importance of this document. You need to have a complete knowledge of environment that you need to operate a particular system of product. That includes the specification of the operating hardware and software, etc. And also there must be an alternative stepwise manual installation, or at least the description of what actually happened during the entire automated process.
- (c) **Introductory manual:** Introduction manual is purely for the purpose to introduce the product to the users. Its integral parts should be the reason why this product has been developed, then all important and lessimportant functions of the product. How to operate to achieve a particular function done. And also other information like where we can get help from, in particular situations. So the primary objective of the document is to get started with facilities provided in the product.
- (d) **System reference manual:** This document is referred as listing of all the error messages you can get during the operation of the system. And how to deal with every single state of error you face. Every time when a user faces an error, he/she needs not only to come out of that state of error successfully but also to get the operation done to achieve the desired functionality. So a stepwise procedure to implement to get the desired job done from different states of error is provided in this document. This document can never be finalized and requires updating for the lifetime of the product.
- (e) **System admin guide:** This document is considered to be as extra information for the administrator explaining different types of behaviors of the system in different environments and with other systems, and how to deal with all those situations. I personally consider this document as part of the installation document. The different behaviors of

the system to different environments should be provided along the installation process information.

3.3.2 System documentation

System documentation is the process of arranging all the documents or information we achieved at the end of each phase. Now right from the beginning when the requirement engineer starts with a reason why we require a system. Then he/she starts collecting specifications for the system with the help of requirement document. Then all the processes and the work that has been done by the designers and architects to provide a sketch of a convenient way of achieving the goal. Understanding of the design, implementation and testing of the system is very important for future maintenance of the system. Moreover, it can help produce similar products in the future. Therefore, for system engineers and programmers the understanding of the whole system, its design, it's essential parts and the functionality of each of the parts is very important. The more detailed design work will make it much easier for the programmers to achieve the functionality with coding. It gets much easier for both the programmer in the process of development and the one maintaining the product later, if you have a thorough and easy to understand design architecture. Traditionally, every single piece of paper produced during this process of development has to be included in the system documentation. What actually needed is to arrange them in such an order and with essential information regarding that piece of work, So that it becomes easier for the reader to understand. Following is the essential parts that system documentation must cover. This is not the final list, but we think it includes all detailed smaller parts.

1. **Requirement document:** This is a whole complete phase during the development process. It is the foundation of the development process, in which the rest of the development process depends. It presents all detailed descriptions of what the user needs are and how can these be fulfilled. The requirement document is the agreed-upon document between the customer and requirement engineer (company). The agreement is on the needs of the user and an idea of what actually will fulfill those needs. This document also clarifies the rationale behind the development of a particular functionality system.
2. **Design and Architecture:** This is a very innovative work, and it provides a whole sketch of how the system can be achieved. The architectural design of the system simplifies how the system will work and how different parts within the system coordinate to work together. Different functional parts must be arranged in a design where it could produce a collective functionality in the most convenient and efficient way. And also this design architecture should be simple enough to understand how the system works. A comprehensive description should be provided that describes how the design works. Let me mention that functionality of every component must be presented and how these functions are produced in the right order and at the right time.

3. **Source code:** Source code produced is included in this part. However, the important thing to mention is that source code should be simple enough to be followed and understood. And that can be achieved by using appropriate relative names while naming the functions and variables, etc. in the coding, and also there should be proper labeling and commenting that explains what a particular piece of code is doing.
4. **Validation, Verification and Testing:** I have arranged these three in the same category but each one of them is as important as any other part. I will not describe here what each one of them is supposed to present, but all results from the process of validation, verification and testing must be provided with sufficient description.
5. **Maintenance or help guide:** Here the dependency of different parts within the system, whether it's hardware or software is presented. Some of these dependencies are crucial for different functionalities. So these structural and functional dependencies must be clarified to make it easier for future maintenance of the product. Mostly certain hardware depends on certain software and vice versa. So this guide is genuinely to present the known problems and the solutions to those problems.

The comprehensiveness of each of the documents discussed above whether it's a user document or a system document depends upon the size of your project. For example, sometimes work done on the documentation takes more time than the actual implementation of the project in some larger projects, while in smaller projects it becomes simpler. However, for all whether it's a small or a larger project, all the above given documents must be provided in some form. However, in different details depending upon the project. The above given parts or components of documentation are not in a much detailed form, but my intention here is to present some very important and crucial components of documentation. The development is of a small system or a larger system. The given components must be addressed in one way or another. The above given is not a standardized set of components but some essential parts of components according to my understanding. That must be achieved no matter which methodology you are using during the development. The number and names of the components may change, but the information is always the same. Now when we have enough information on the documentation, we would go into a description of the agile methodology.

Chapter 4

Agile Software Development

This chapter first describes Agile manifesto, and then it explains few agile claiming methodologies and their functions. Methodologies include Scrum, Extreme Programming and Feature Driven Development. It also discusses that how these methodologies relate to documentation.

=====

4.1 Agile Software development

Forty years of traditional software-development methods play a vital role in software industry, although these methods were rigid, well-planned and provide a little amount of flexibility. These methods were not provided to accommodate changes in the late phases during the development process but with the passage of time, more software-development methodologies introduced and customer demanded more flexibility in terms of changing requirements [Biju \(2010\)](#). These methodologies captured vast majority of the market and almost 69 percent organizations using agile approaches on one or more projects [Ambler \(2008\)](#). It is also a popular research topic in academia. Agile means an iterative approach with a strong focus on goals and more customer involvement [Prause and Durdik \(2012\)](#). Software practitioners and software developer recognize that there are some drawbacks in spiral, incremental and other traditional development methods. So in 2001 a group of 17 software developers get together in Utah and formulate a new method under the Agile manifesto. Agile manifesto consists of four basic principles.

Following are the main four points in agile manifesto

- Individual and interactions over process and tools: This principle gives more attention to individuals who participate throughout the development process, and it does not focus more on process or tools. This principle needs to be more specific to interaction and communication among team members and customers.
- Working software over comprehensive documentation: This principle states that the main aim of the agile development should to produce a working and quality product. The main

purpose and focus will be on the actual code and give less attention to huge documentation, but only necessary documentation will take place.

- Customer collaboration over contract negotiation: This principle mainly focuses on customer, that how a customer plays an important role in software development, and it emphasize communication between customer and development team. It encourages the software developers to base their work on going in a contract with the customer, so the more efficient communication takes place between software-development team and customer, the more will be a good-quality product.
- Responding to change over following a plan: This last point of agile manifesto encourages the software developers to establish a process that can handle changes successfully during the development process without compromise quality of the end product. Customer cannot predict all requirements at the start of the development, so this principle gives a facility that customer will understand requirements during the development process, and then he can share at any stage with the development team [Fowler and Highsmith \(2001\)](#).

There are few agile methodologies (Scrum, Extreme Programming, Crystal, Feature driven development and dynamic system development. We will discuss scrum in following paragraphs.

4.2 Scrum

Among these methods scrum and Extreme programming play an important role and using in today software development. Most of the time they work together but scrum focuses on the software project while XP focuses more about engineering practices. Scrum is a simple framework and invented by Sutherland's team at Easel Corporation in 1993. Scrum which is to be used to organize team and get work done to be more productive with good quality. It allows team and team member to choose the amount of work to be done and decide how best to do it. Therefore, it provides more enjoyable and productive working environment [Chen et al. \(2007\)](#). Ksenya Mizinova is a technical writer who depicts it in her own words as "In Scrum approach, it's assumed that technical writers are members of the development team thus they are "pigs" (in terms of "pigs-chickens" classification "pig" is a person who is committed to the project". However, Scrum gives more attention and focusing on prioritizing work based on business value, improving the quality and usefulness of what is delivered and in this way increase the revenue. Scrum development takes place and adapts in such a way that it will be flexible to changing requirement during the development process at short, regular intervals. Scrum can allow a team to prioritize customer requirements and change the work product in real time to customer needs. By doing this scrum to know what the customer wants at the time of delivery and in this it improving customer satisfaction while eliminating waste work that is not highly valued by the customer. Scrum (uncertainty Scrum) is one of the agile processes used by small team at Altitude Software to manage the process of writing user documentation. Scrum allows the team to effectively prioritize regular work along with the

difficult creative work. “It overbooks writers on iterative cycles called sprints, and then lets the writers micro-manage their tasks to overcome obstacles. After each sprint, the team decides what to publish and whether to proceed with unfinished work” [Baptista \(2008\)](#). Scrum is simple “inspect and adapt” method and software-development process designed to add energy, clarity, focus and transparency to project teams developing software systems. A properly implemented scrum was designed to increase speed of development, align individuals and organization objectives, create a culture driven by performance, support shareholder value creation, achieve stable and consistent communication performance on all levels and enhance individual development and quality of life. Scrum is a process mainly focuses on iterative development. It has three main roles, three documents and three meetings. Three roles of product owner, team and scrum master.

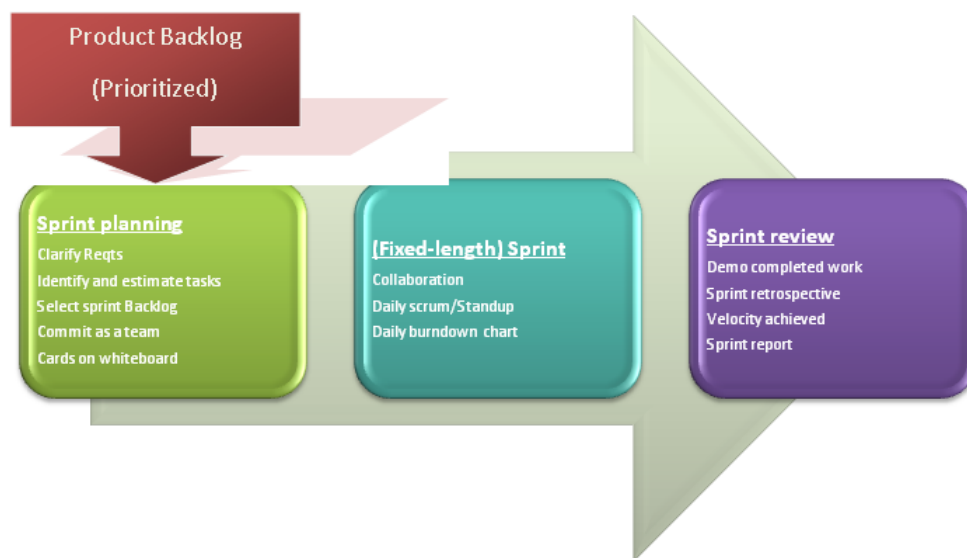


Figure 4.1: Scrum

4.2.1 Roles

1. **Product Owner:** Product owner has a responsibility to prioritize customer needs and represents stakeholders such as customers and marketing, etc. It is the product owner who is responsible to approve and reject requirements for the project.
2. **Team:** Agile software development needs two to eight members, and team need to be cross functional. Scrum team should need to organize itself and its work. Team must inform the scrum master if any hurdle and hazard face by the team.
3. **Scrum Master:** Scrum master responsibility to promote development team and remove all hurdles during the development process. He can tie bond of good communication between team members.

4.2.2 Meetings

Three meetings are the sprint planning meeting, daily scrum meeting and sprint review.

- **Sprint planning meeting:** At the start of each sprint, sprint planning meeting takes place. In this meeting scrum master and product owner discusses product backlog, fix goals to be gain in current sprint and give insight to understand thinking of the product owner. Scrum teams takes a task from the product backlog and makes a commitment of completion of the given task within the specified time.
- **Daily scrum meeting:** Member of the scrum team meets daily in the morning for fifteen minutes, and everyone from the team attends this meeting. It gives an opportunity to the team member to participate and provide their progress and hurdles they faced during their development process.
- **Sprint review:** At the end of each sprint, sprint review takes place. In this meeting, team member present and demonstrate what they have done during the sprint. Management, product owner, scrum master take participation at this meeting. This meeting last from few minutes to few hours and get feedback what have been done during the sprint [Schwaber \(2004\)](#).

4.2.3 Documents

Three documents are product backlog, sprint backlog and burn down chart.

1. **Product backlog:** Product backlog contains a list of prioritized items. Items having highest priority can be broken down into small manageable that they can be forward for further estimation and testing.
2. **Sprint backlog:** The scrum team selects top priority items from the product backlog and put it another backlog called sprint backlog and scrum master makes a commitment of delivery within a specified time.
3. **Burn down chart:** During sprint planning meeting sprint team point out and make a commitment of a specified task from the sprint backlog to be completed in a given schedule time. Burn down chart is a graphical representation between the work left out and duration of the work. This type of work usually created in excel, share point and white board [Deemer et al. \(2010\)](#).

4.3 Extreme Programming

Extreme Programming is one of the agile methodologies designed for medium team and work well in a rapid changing requirement. From the name it is clear that this method is a program centric and gives more attention to technical practices for skillful development through frequent

delivery of working software [Kuppuswami et al. \(2003\)](#). Two of the main characteristics which XP distinct from other methodologies is cycle time and level of ceremony. XP always recommends short iteration from one to four weeks. Few artifacts in an XP project like story, code and unit test. XP is well known for their potential benefits by improving software in terms of fewer bugs, early delivery of valuable functionality, maintain interaction between user and developer and try to low the curve of cost/change (Andrew Jackson et al). XP project usually takes time from six to fifteen months, and team consist between two to twenty members. Most of the time XP depends on oral communication, tests and source code to communicate system structure and intent [Briand \(2003\)](#). XP follow the following four core values [Kobayashi et al. \(2006\)](#).

- Communication
- Simplicity
- Feedback
- Courage

XP also defined several practices, and the use of these practices depends upon the availability of the resources and also depends upon the nature of the project. Following are the twelve practices.

- Small releases
- The planning game
- Metaphor
- Simple design
- Continuous Testing
- Re-factoring
- Pair Programming
- Collective Code ownership
- Continuous Integration
- 40-hour week
- On site Customer
- Coding Standards [Hunt \(2006\)](#)
- Extreme Programming and Documentation Traditional software-development methods were heavy weight, rigid, heavily documented and were process oriented. XP introduced pair programming in which two Programmers/Developers set side by side on one computer. In

this way, they communicate easily throughout the project, and it reduces a lot of details later in documentation development. The pair programmer involved in intensive dialogue, share their views, together they to design better, develop and test a program during the whole project development [Wong and Tilley \(2002\)](#).

4.4 Feature driven development

Most agile methodologies start with a list of principles or present some set of processes while FDD gives more attention to the domain model. Creating a model for FDD is the very basic step and for that collecting domain knowledge from all domain experts should be required and then integrate into a unified model representing a domain model. FDD composed of five specific processes and should be followed in a following specific order.

- develops an overall model
- Build a list of features
- Plan by feature
- Design by feature
- Build by feature

FDD always focuses on roles and responsibility more than other agile methodologies and further more FDD tries to away from shared ownership of code as XP and other agile methodologies promote it. Here are the nine defined roles in FDD.

- Project manager
- Chief architect
- Development manager
- Chief programmer
- Class owner
- Domain expert
- Tester
- Deployed
- Technical writer

FDD uses unique and specific mechanisms (feature list and task list) to report project activity and progress report. FDD defines six mile stones for each feature.

- Walk through
- Design
- Design review
- Code
- Code review and unit test
- Promotion [Chen et al. \(2007\)](#)

Chapter 5

Literature Discussion

In this chapter, we took data from a literature in which opinions about Agile documentation of some popular researchers and authors have been presented. We also used some data from a very related research work done on how internal documentation is important and that what team members consider documentation at work and their project. This data is shown in the form graphs.

=====

5.1 Literature Discussion

Now one of our research questions is that "What is the rationale (motivation) to produce and use documentation in Agile development?" Different renowned Agile experts have different views, and they also have done work on it, which we would like to present here so that could have a better understanding on the matter, also if we could find answers for our questions. Now just to remind you that if you search it, you will find much work done with the title of 'Agile verses documentation-driven methodologies'. This actually gives the impression that if in Agile methodology, documentation is of less importance. So this could be one of the reasons why we think there is a need to clarify the need for documentation (If there is any) and the importance of it. About a decade ago due to the dramatic growth in the use of Agile methodology, researchers from four institutions Fraunhofer Center for Experimental Software Engineering, North Carolina State University, University of Southern California and University of Maryland organized an e workshop to discuss and get benefited from the experience of experts. Eighteen Agile experts spread across the globe were invited. When they were asked about documentation in agile methodology, here are some of the answers from few of them. Scott Ambler says that documentation becomes out of date and should be updated only "when it hurts". And that sometimes it is necessary in order to retain critical information over time [Ambler \(2002b\)](#). Barry Boehm says that a documented project makes it easier for an outside expert to diagnose problems [Boehm \(2002\)](#). Bil Kleb said that "with Agile Methods, documentation is assigned a cost, and its extent is determined by the customer (excepting internal documentation). Agile method proposed an approach to software development

that eliminated the necessity of documentation. They claim this by doing informal communication between developer and customer, code standardization and pair programming. So it greatly reduces the need for documentation in software-development [de Souza et al. \(2005\)](#). Documentation should be correct, consistence and complete so then it can describe the software processes and systems in a systematic way, and the reader can understand and pick an idea effectively. On the other hand, poor documentation is the basic reason to degrade the quality of the software system [Kajko-Mattsson \(2005\)](#). Agile is a people-oriented methodology and communication play in an important role in software development, but it does not mean that written documentation should be avoided. However, too much documentation is expensive and time-consuming to produce and read [Mazni et al. \(2010\)](#). While the documentation process should be conducted in such a way that on one hand it is feasible to stakeholders with various backgrounds, and on the other hand, it shows the benefits of their time spent on sharing knowledge. [Buckl et al. \(2011\)](#). Nokia also improving their documentation work without compromising their quality of documentation. Method through which the documentation has been developed is called RaPiD7 (Rapid Production of Documentation in 7steps) presented from Philips Digital System Laboratory. This method is now used outside Nokia too, and first trial of the method used by Philips during April 2004. The most important benefits get through RaPiD7 methods is that it works more on agile way and gives more attention to the following points. i. There is more interaction among project stakeholders, project members, customer representative and third parties if involved. ii. Focus is only given to essential documentation iii. Create common understand among share holders to transfer knowledge effectively iv. Need to create more focus, which should results in quick results and output v. Also respond to changes effectively. The above benefits were expected from the RaPiD7 [Dooms and Kylmäkoski \(2005\)](#). "Ambler suggests two primary reasons for documentation, namely that we should model (or document) to communicate, or model (or document) to understand. He also recommends that the documents to be produced to support the thinking associated with each stage during the project. In other words, producing the document develops and supports the understanding. Document driven approach suggest that the documentation produced should be in line and will work as a natural byproduct of the project rather than as an afterthought hurriedly pieces together at the end" [Clear \(2003\)](#). "Producing usable documentation has always been a tough and hard task, and communicating important knowledge about a system among collaborators is difficult. Agile experts and practitioners often stress the fact that documentation is time consuming, hard to maintain and usually does not provide great added value to the working software. Documentation must be light, because the system is constantly changing, and it is best to spend most of the time on writing test and features than updating documentation. It is better that documentation need to be near to the source code as possible and also need to be light, create quick and mostly free to maintain" [Dagenais and Ossher \(2006\)](#).

Software-development organization needs evidence that agile method can suit and apply in their working environment. Therefore, software engineering researchers conducting empirical studies from industrial point of view and provide more empirical evidence. The Agile Researchers always gives more attention to agile perception and comparison studies. Documentation plays

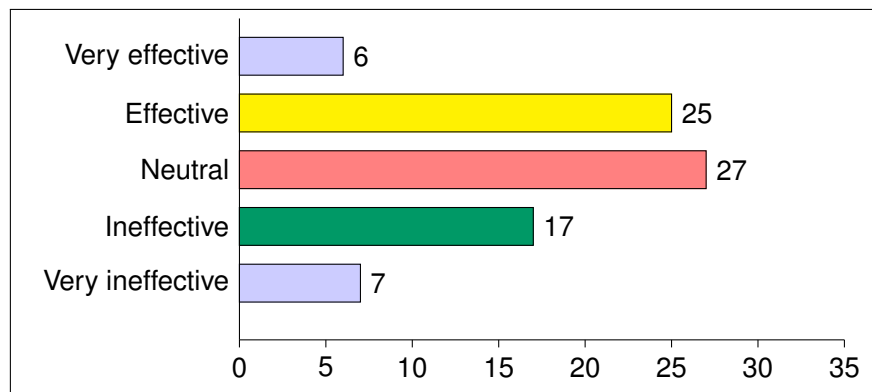


Figure 5.1: How effective do you consider finding internal documentation? [Stettina and Heijstek \(2011\)](#)

in an important role in a software-development process. It is considered as a model and communication key between product and customer. In the development process, we cannot neglect documentation but at the same time too much documentation is cumbersome and hard to produce and read. Therefore, in the agile manifesto, it is clearly mentioned in one of the principles that states, "working software over comprehensive documentation". Agile team always trying to focus on the actual product under development rather than to make long documentation.

[Stettina et al. \(2012\)](#) presents an empirical study of internal documentation in agile software-development teams. We think this study is very relevant to the subject, and helps a lot in understanding the problem domain and the results obtained in their survey is also very helpful in a fruitful analysis of the domain. We will also give an overview of the questions they asked and the results they received.

According to the authors, they wanted to investigate the role of documentation in agile development teams. So they employed a questionnaire to measure different perceptions of a group of agile developers (practitioners). Now they received responses from 79 professional individuals distributed in 8 teams from 13 different countries.

A two-part questionnaire has been developed. Part one was for collecting the perceptions of team members regarding their documentation. While part two was designed to inquire about the software tools applied to manage that documentation. Now some of the findings have been depicted in the form of a graph given below [Stettina et al. \(2012\)](#).

In the Figure 5.1 the question asked is that 'how effective do you consider finding internal documentation?' internal documentation is the most controversial thing in the agile way of working. As according to some practitioners, the methodology focuses on eliminating internal documentation. But the results from this sample survey show that a majority of developers thinks it's effective or neutral. But they are not saying it's ineffective or very ineffective. So here is a positive signal for internal documentation. In Figure 5.2 the question asked was that 'How do you feel about documentation at work?' Now keeping in mind that they are asking these questions from 79 developers divided and working in eight different teams from 13 distant countries. And they are agile practitioners. While the results they have provided are that almost half of the developers think of

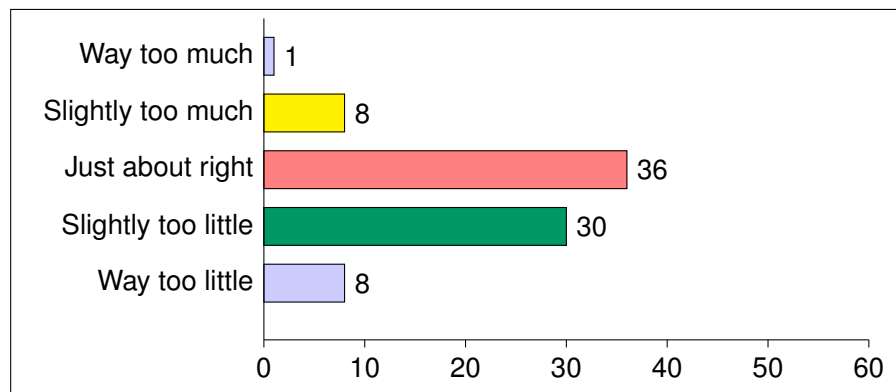


Figure 5.2: How do you feel about documentation at work [Stettina and Heijstek \(2011\)](#)

their documentation to be too little. This means they want the documentation to be either more detailed or they think of the current documentation as insufficient. In Figure 5.3 the question asked is 'How important do you consider documentation for your project?' now we would expect that agile practitioners would rate documentation as not so important but the results for this question are remarkable. As majority of participants defines documentation to be either important or very important. The results here clearly define how important documentation is considered around the world among practitioners.

As presented in Table 5.1, the perceptions on the amount of documentation, effectiveness in finding internal documentation and importance of artifacts are accumulated from the data. According to the respective Liker scale, the values are distributed along a scale from - 2" to +2". A 0" thus corresponds to just about right", a +1" to slightly too much" and a +2" is way too much" while a -1" and a -2" correspond to be slightly too little" and way too little", respectively. The value of - 1" therefore means that the team perceives documentation as slightly too much", while the value of - 2" would represent a team perception of documentation as "way too little". Consistently, with the global sample Table 5.1 reveals that more than half of the researched teams reported perceived a deceit in the amount of documentation. Note that none of the teams perceive their documentations

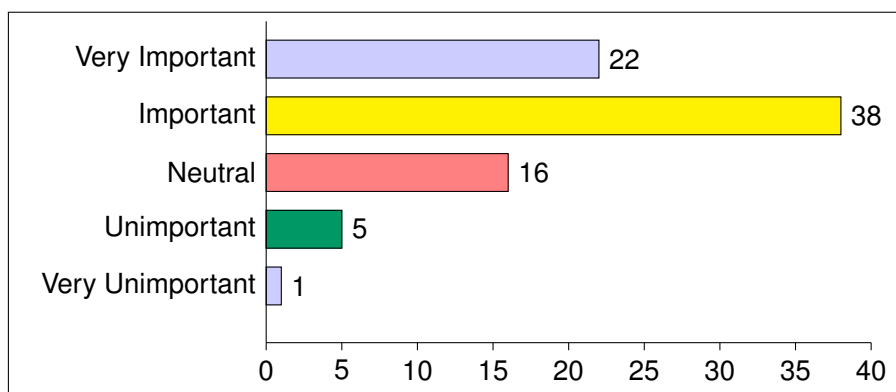


Figure 5.3: How important do you consider documentation for your project? [Stettina and Heijstek \(2011\)](#)

as "too much".

Table 5.1: Descriptive variables team results (χ) and agreement(σ^2) Stettina and Heijstek (2011)

	T1	T2	T3	T4	T5	T6	T7	T8	Avg. Agr.
Country	UK	US	UK	NO	NL	SE	IN	NZ	
Team size (pers.)	4	9	5	12	6	4	8	6	
Collected answers	4	6	5	6	5	3	8	4	
avg. exp. (yrs.)	7.75	13.7	6.6	12.7	2.6	10	7	3.5	
special distribution	co-loc	co-loc	co-loc	distrib.	co-loc	co-loc	distrib.	co-loc	
documentation tool	Wiki	confluence	Wiki	-	-	Wiki	Confluence	Wiki	-
perceived document amount	x	-0.25	-0.5	-1.3	-1	-0.75	-0.13	0	
	σ^2	(-0.19)	(-0.25)	(-0.89)	(-0.4)	(-0.67)	(-0.61)	0	-0.56
perceived eff. .	x	0.65	0.76	0.6	0.52	0.5	0.75	0.45	
finding document	σ^2	(-0.69)	(-0.47)	(-1.33)	-1.44	-0.89	-0.69	-0.69	-0.8
perceived important artifact	x	1	0.7	0.57	0.72	0.75	0.7	0.85	0.72
	σ^2	0	(-2.25)	(-0.47)	(-1.04)	(-0.67)	(-0.5)	(-0.69)	(-0.72)
average agreement	σ^2	(-0.29)	(-0.99)	(-0.9)	(-0.96)	(-0.74)	(-0.6)	(-0.46)	(-0.69)

Chapter 6

Data Analysis and Interpretation

This chapter present, analyze and make interpretation from the obtained data. The data is mainly taken from the discussion forum, in which every practitioner shares his/her views about each component of the documentation. In the last of the chapter, we have concluded result in the form of factors that affect documentation.

=====

6.1 Data Need

Now why did we feel the need to get this data, is because no matter how much you study the literature and different theories presented by individuals. It does not provide us with some factual answers that we need. For example, what parts of the documentation are practiced by professionals? For that purpose, it was important to have some data collection from people related to the field. Who not only have knowledge of the area but also related to similar type of work or are currently working on projects implementing agile methodology. So the questionnaire has been developed consisting all the essential elements. Every single part of this questionnaire has questions in it. These questions are related to that particular portion.

6.2 Data Analysis

The total number of categories is 9, and we have a total of 34 questions in this questionnaire. Now the number of questions differs from one part to another. These questions that we asked depended on the kind of information we wanted to know about that particular category. Now we will be discussing all the nine components below categorically. To support the analysis and discussion done along with each part, a graph will be given along. Now these graphs we have generated from the data that we collected.

6.3 Documentation Components

6.3.1 User Documents

User documentation refers to the software product or service which is used by the end user. Many software companies and software developers always try to develop user documents for the end user. The reasons to develop user documentations is to build company image, decrease support cost, to help the user while using software, and it can also be used as a marketing tool. Most companies think that if user documentation is not getting any profit so don't create it. User documentation will be not useful until it answered the questions people ask. Good manual always helps the users that how the software or product operates. Following are the most important user documentations.

- Reference manual
- User guide
- Support guide
- Training material

Reference manual is a document which is designed alphabetically for experienced users. Training guide teaches with very basic instructions that how to interact with the software or product, like how to open a particular menu and then different options available under that menu. While user manual or user guide for those who already understand the basics of the software that how to use the software. So this gives detailed instructions to the users about each button and dialog box. User manuals are usually developed in synchronization with product development time line, and it should not be developed under the pressure of shipping deadlines.

Brett Maytom says that

"Create documents that will be used and the live way beyond the project life. Don't create documents that add no long-term value."

Further Alejandro Valdecasas more elaborate the user documents and he says that

Remember, as agilists, we favor working software over comprehensive documentation, but that doesn't mean we can't create the documentation we need. Even more importantly, when we know something is important, it makes sense to write it down.

The documentation that must be developed to support successful deployment and use: 1. The user manual. 2. Online Help 3. A document that includes installation instructions and configuration guidelines is sometimes necessary. Furthermore, a "read me" file is included as a standard component. 4. Labeling and Packaging"

Chris Johnston says that

Create code that is self-documenting and create easy to use tools that do things like installs, updates, etc. Create software that is intuitive and helps the user do what is right and makes it hard to do what is "wrong". Use User-Centered design and include your customer in the creation of the software and you will find that you need to create a lot less documentation. I am not against creating documents, but only create the ones that are absolutely needed. I find a lot of times, documents take the place of creating good, clean intuitive software. Development teams would much rather have someone else create a user guide instead of taking the time to create the software right.

So, bottom line, don't ask us what documents to create, ask your users and your support people and then either change the software/tools or write the guides.

Allen Holub says that

The sorts of documents you list are user-level, not internal, documentation, so the usual rules about self-documenting code don't apply. However, agile systems tend to change constantly, even at the user-interface level, which is to say that this user-level documentation will also become stale very quickly. I'd suggest skipping all the written documentation. What you do need is:

- * An intuitive user interface that maps easily to the user's notion of the domain. This one is essential. If you need a "reference guide" or a "training manual" to explain how the program works, you've done something fundamentally wrong. An intuitive interface should be a side effect of working from user-supplied stories, but a dysfunctional shop (where stories come from a Marketing dept. rather than actual users, or where the system architecture does not map 1:1 into the domain) might have a hard time with it.
- * Context-sensitive help, ideally intelligent context sensitive help that figures out that somebody is struggling with some task, and then helps them. The help should be integrated into the code so that it's easy to change it when the code changes, and the architecture has to support that integration. If the help text is all in a separate "help" subsystem, it will become as stale as the paper documents.

Things like installation guides are occasionally useful, though an installer program that automates the installation process is better. Think of how programs like Homebrew have more or less eliminated the need for complex installation guides in the Mac/Unix world.

NK Sharivastava says that

My team produces the documents that are required by the customer and are included in SOW or contract. Other than that they don't create a lot of documents. The two that they like and create are functional/business requirement's document and test case documents.

6.3.2 Design Decision

Agile is the only methodology which is known for its non-documented designing. It means that some or most of the agile practitioners think that the design process in an agile way of working is most often not documented. And even if some sketch work is done, it is temporary. Now as we all know that in a traditional way of development, the design decisions are taken by the software architect. So the role of the software architects in agile teams and the status and work done by him/her also is of importance here. The subject has been raised on a forum with thousands of Agile practitioners and professionals from highly experienced ones to the beginners. Now there are different responses to the subject. However, as it is a continues discussion with different perceptions and thoughts from different practitioners. So we would just include some thoughts and their experience from some of the practitioners below. As Chris Johnston says that

```
there is no role of an architect in an agile team
if they are not creating code on the daily basis,
and that you just need to have a high-functioning
development team with a senior developer or two.
Chris thinks that the people closest to the final
solution should be the ones making the decisions
regarding architecture, libraries, gems, protocols,
etc.
```

we noticed that this comment was liked by many members, and also many of them agreed with it in their comments too. If we read it carefully in this comment role of the an architecture has been completely eliminated. Who actually is responsible for those detailed architectural designs. And its been suggested in the comment that an organized group of developers is all needed in an agile way of working. And that the experienced members who are close to the solution are the one, should be making design decisions. So this comment identifies the lack of experience and is related to experience factor.

Another respondent, Valentin-Tudor Mocanu suggested Robert C. Martin considerations about agile design in “ Agile Principles, Patterns, and Practices”. Valentin also says that the real design is in the code and that the code should be good enough to have all important design decisions visible. However, he also says that there is no “no documentation” approach in agile, but rather a “document essential” approach. These comments can be related to two factors, the product type and the tools used.

```
The agile way should be opportunistic depending upon
the context. There is no 'no documentation' approach
in agile (that is rather an anti-pattern), but rather
a 'document essential' approach. There are many
techniques such as 'Agile Modeling', but the context
is what should be considered.
```

Paul Oldfield is another experienced respondent who is saying the same thing that.

The source code is what is of high importance and everything else is optional and depends upon the team and the context.

Similarly, Tristan Gutsche also responds that.

There is no formal document for design decisions that he is aware of and that it all depends upon the team and the context. He says that there was only once that he had a living diagram that showed the progressive state at the end of each iteration, and he says that it was all they needed to present to show the issues identified, and the technical solution provided. According to him, the importance of this document was communication with all project team members. He found it really effective for ongoing negotiation and alignments. The decision changes were all in a box on the diagram. It forced them all to be no more than 2-3 sentences.

Now if we observe, Tristan's statement also states that it depends upon the type of project you are working on. For instance, that one project when he had a living diagram, he termed it to be very effective for ongoing negotiation, alignments and communication. Still he never did it again, that means that he was okay working without it in other projects. Now this could be because of the type of product he was working on, it could also be the size of the project. Now results that we received from our survey also has a similar kind of results, as shown by the graph below. Three different questions were asked relating to design decisions. The first one was, do you think the design decision is important to be documented? And 9 out of 10 respondents says yes, the second question followed was, do you practice system and architecture designs in your development? Now 8 of them said yes we generate these designs, while 2 said no, we don't. The third question relating to design decisions was, do you practice documenting those designs? 7 practitioners said yes we document it, and 3 of them said no we don't document it.

6.3.3 Vision statement

Now this document is for the Senior Management, User Management, Project Management. That means this vision statement is confined to the managerial and executive level. The document consists of the definition of the vision for the system and a summary of the current estimated costs, the predicted benefits and the scheduled milestones, etc. Now when we put this component on the discussion forum to find out what the practitioners think about it. We find out that there are different versions of this document. Like some called it project charter, some called vision statement or executive Overview and some even referred as a business case. However, the most

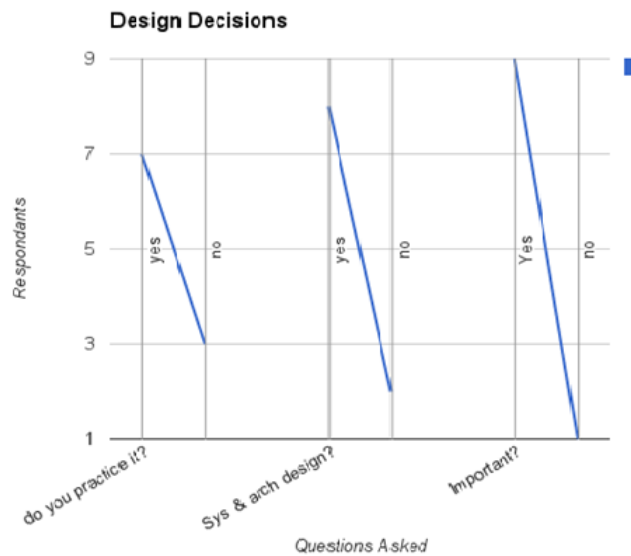


Figure 6.1: Dsign Decision

popular one is the vision statement. Most of the individuals understand it much clearly when you call it vision statement. Now as we said that different practitioners use this information in different set of arrangement and call it with different names. Following are some of the comments that we are going to present here, so that you could understand the use of this information in different forms in the market. It depends a lot on the organizational culture. We prefer to have both the business case and the charter in place for most of my projects. At least to make sure everyone understand the goals and to get the official commitment from the sponsor. However, let me tell you that charter too is not as such a defined set of information. It may change from person to person, also the type and amount of information may vary from one project charter to another project charter. The reason could also be the project, but mainly different persons arrange their own set of information they require as a project charter. So we would place this statement in the environment section of our effect categorization structure.

David Bland's statement is that

```
We may want to define the project charter a bit
because it ranges from what I'm guessing Tom De Lancey
has experienced (and others) to what Ainsley and Diana
wrote about in the book Lift Off.
Chartering in an agile context to me is something like
Vision Statement -> Business Model Canvas -> Empathy
Maps / Personas -> Experience Map -> Working Agreement
-> Backlog -> Board.
```

David's comment shows that he and the other two people who liked the comment, thinks that

agile starts from the vision statement and charter would be more than a vision statement. Which we too think is right, but different people requires a different set of information at the beginning of the project. So a project charter could be a vision statement along with some other required information.

Paul Oldfield says that

I like to have business case and vision statement - the vision statement tells me everything that i need to know that might be on the charter; the business case gives me a first look at what is valuable to the business.

So Paul likes to have a business case along with vision statement.

Another respondent Kaushal Gohel says after agreeing with Paul about Vision statement importance and then with Tom's statement that

Another respondent Kaushal Gohel says after agreeing with Paul about Vision statement importance and then with Tom's statement that " it is useless because when a project kicks-off, everyone knows what it's all about." Further he highlights Tom's point that is

Agree with two opposite views mentioned above. First from Paul that vision is important. It gives bird view of whether carvan is going in right direction or not. Second, as Tom said, it is useless because when a project kicks-off, everyone knows what is all about. However, Tom himself mentioned one interesting point to keep the vision statement. The question that whether this task will help the coders? A project is a journey and there is a possibility of few changes in the team during the journey. A well-defined vision statement is helpful for any new person joining the team to understand whether the construction is for a castle or for a chateau. Surely, it doesn't give complete understanding but that is also not the intention of a vision statement.

So a project charter, just containing a well-defined vision statement is good to have. However, as they say, don't burn ocean for a handful of salt. Like every single task of Agile, time box task to define vision statement so that you don't loss valuable time just in trying to prepare a project charter.

That is a well-defined vision statement is helpful for any new person joining the team to understand whether the construction is for a castle or for a chateau. This statement shows that the experience of the team also affects your development, and that a more experienced team won't need that much detailed documentation as a relatively inexperienced team would require.

Dan Williams has decades of experience as a developer, he says

I believe a charter or vision statement can be a very important and effective communication tool for team and stakeholders. One or two pages should be sufficient to cover initial scope, budget, schedule, agile framework approach, reporting, initial risks, team composition, etc.

This can be called a perfect statement except one point that we would comment on, is that the vision statment size in terms of pages depends upon the size and scale of the project. These two are directly proportional to each other, the bigger the project and the bigger the size of the vision statement would be.

NK Shrivastava says that






In my view Project, Charter is a formal tool to define and approve a project. It depends on organizational processes and culture whether a project charter is needed or not in a formal manner, but every organization would need a project definition (objectives, beneficiaries/stakeholders, scope, time, cost and risks) and someone needs to approve before the project can begin. If that is the case, then every project would need to have a charter formal or informal.

So this statement shows that it also depends upon the environment you are working in, as different companies have different environments and ways of doing things.

The Product Vision Board is one very popular way of storing and presenting this information. We just to give an overview of this product vision board. It has five sections explained by the following diagram. Vision board¹

- Vision Statement: It is a concise summary of the idea that actually describes what you intend to achieve with it.
- Target Group: Now this portion gives the beneficiaries of the product. The customer and the different users could be part of this segment.
- Needs: It describes the needs of the users and the customers. Which the product must meet.
- Product: It summarizes the key features that the product will provide. The key 2 to 5 main features that must address the needs of the customers.
- Value: It shows the desired business benefits. And explains why it's worthwhile investing in this product. The number of columns may increase according to the needs of the statement writer. The following approach is commonly suggested approach to identify the risks and the attack those risks.

¹www.romanpichler.com/blog/product-vision/the-product-vision-board/

Vision Statement  Crisp summary of the vision / idea.			
Target group  Which market segment does the product address? Who are the target users and customers?	Needs  Which needs does the product fulfil? How does it create value for its users? Which emotions will it evoke?	Product  What are the three to five top features? What are its unique selling points?	Value  How is the product going to benefit the company? Will it, for instance, increase revenue, enter a new market, develop the brand, reduce cost, create valuable knowledge?

<http://www.romanpichler.com/>



Figure 6.2: Different types of user documentation

Above given was some perception of different practitioners of using vision statement, then a popular format of a vision statement used in the market. Now following is the Fig 6.4 presented, which describes some questions and the responses we received to those questions. We asked 3 questions from 14 different practitioners. The first question was that do you practice vision statement so 7 of them said yes, and 5 five of them said no. However, that does not mean that didn't require the information used in vision statement, and we noticed that different developers use that information grouped in different type of documents. Similarly, the second question was that, how often the schedules in the vision are met? And in the responses we received 1 respondent responded with never, 7 said sometimes and only two said always. The 3rd question was that how flexible usually the schedules are? Now the responses here where that 1 respondent said not flexible, 4 four said less flexible and 5 said more flexible. Now as we mentioned above that the following survey was not meant to get accurate figures, but just to understand the trends and help get answers to the questions raised above. Answers to the second question show that a clear majority met their schedules and milestones sometimes. Which means that setting up the schedules

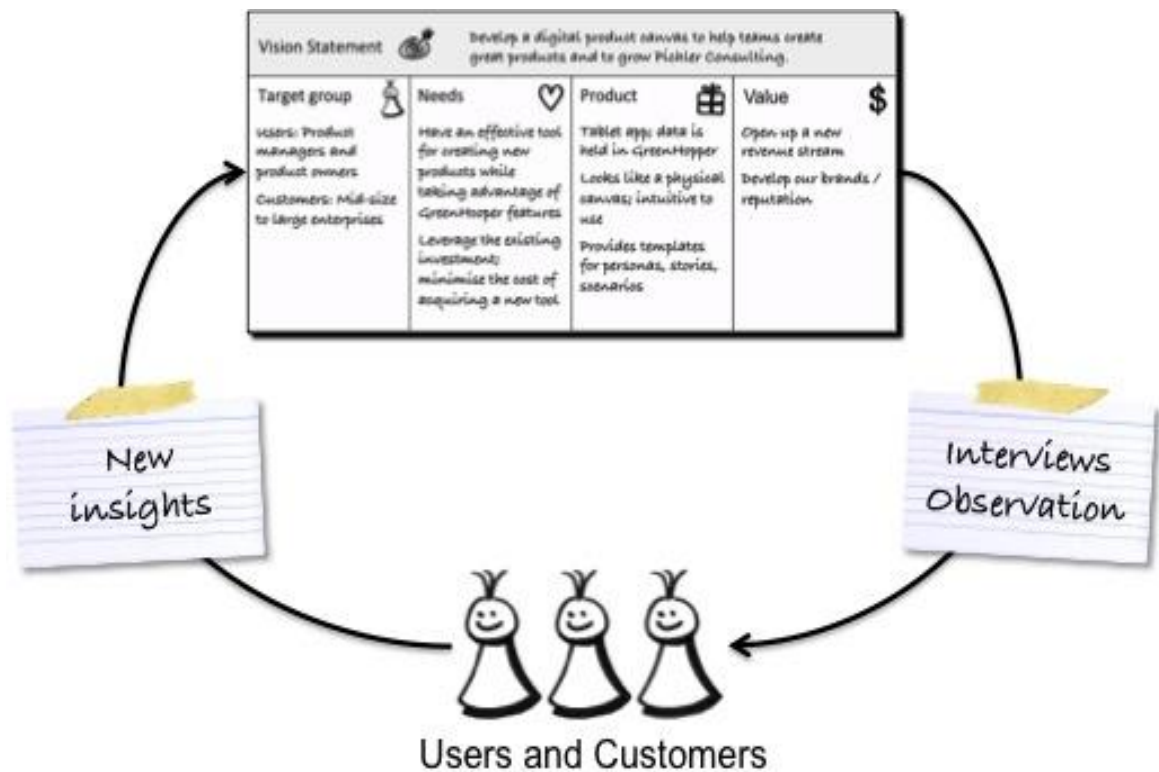


Figure 6.3: Identify Risks and Attack

and milestones do have effects on the overall development. Only 1 respondent said that schedules never met, so this confirms that majority of the developers are getting benefits of setting a vision. Furthermore, if a vast majority of the respondents says they sometimes meet the schedules, that means they do set up schedules and milestones. Similarly, if we analyze the results of the third question, five respondents say they are more flexible, 4 says less flexible and only one responds not flexible. Now these results show that nine out of ten respondents believe that these schedules are flexible, besides the fact that some think less flexible and some say more.

6.3.4 Project Overview

Project Overview is a summary of all Important information, for example, the vision for the system, primary user contacts, technologies used to develop the system and important operating processes. Now Project overview is sometimes misunderstood with vision statement, and from the experience and discussion of different practitioners, we came to know that sometimes the project

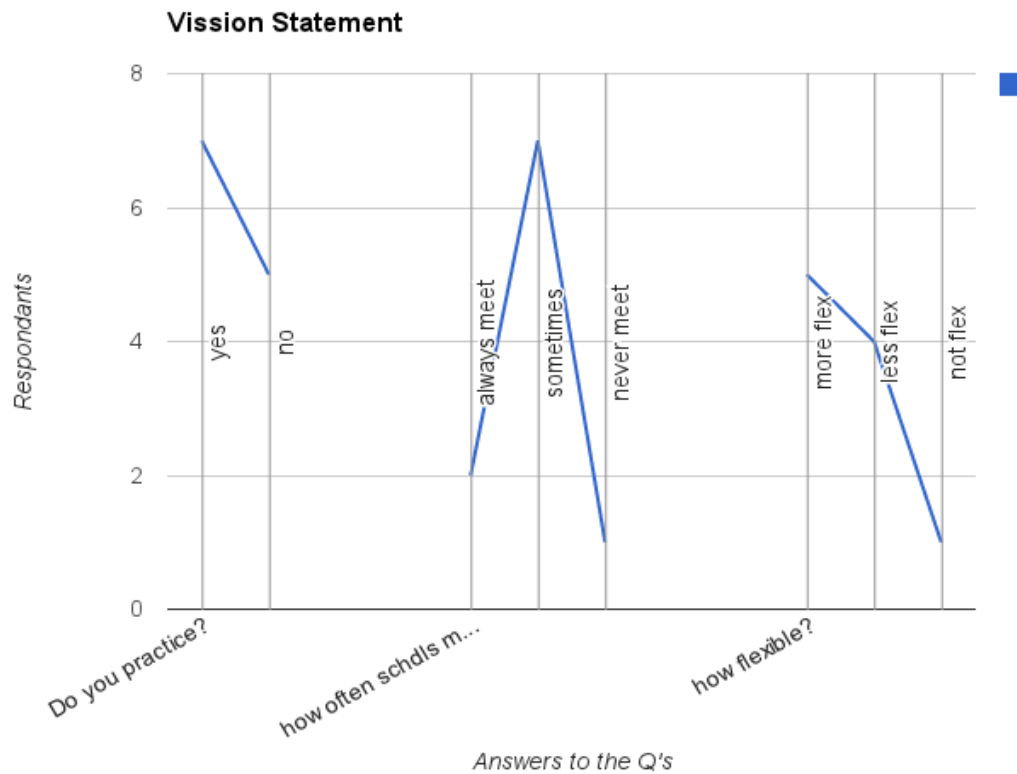


Figure 6.4: Vission Statement

overview is a whole document that also covers the Product (i.e. the important components or the processes the product performs). It's like a complete Project+Product Overview document. The extent of the information provided in the document as we all know that varies from company to company and project to project, and it was verified from the discussion too.

Paul Oldfield says

The Project Overview document vary depending upon the context. I am more likely to have a collection of short documents that one could (if desired) bind together into a 'Project Overview'. The problem, of course, is that different readers would want different topics to be covered.

This binding together different chunks of information and generating a complete project overview document is practiced by many individuals and relatively smaller development companies. This

may also include documents like the vision statement and business case etc. let me say that the perception about the documentation in the market is much different than that of the academic world. .

Brett Maytom says that one should develop Project Overview,

The same way you did before agile:: grin. If the old way did not work, then do it the way it works for you.

It's true that there is not much different between agile or any other way of development methodology when it comes to project overview. Most of the developers are using same tools. However, this very much depends upon the tools you are using, both for developing the product and developing the documents.

Allen Holub's comment is that

The best tool I know is a physical story board that shows all the stories in the system moving from "Percolating" to be done, with the stories arranged by epic and by priority.

This is another good example of how the tools that you use affect your documentation.

Valentin-Tudor Mocanu says that.

Project Overview is used by RUP; The Vision document. This artifact is created in the Inception Phase, and it is important on that time of the project. Anyway, you could check the proposed content. It is an alternative.

Another topic is "Product Overview", that mean (software) system description - with main architecture aspects. For that part, you could check the standard IEEE 1471.

Anyway, for an agile process, there are all the concerns that could be interesting for another process approach. However, It should be captured in an opportunistic manner. Think also to the problem described by Paul: give to any interested reader an appropriate view.

Valentin is also referring Product Overview along with project overview, and second's Paul's idea of binding together different chunks of required information into a Project Overview document.

6.3.5 Requirements Document

Requirement's document defines what the system does. It consists of artifacts such as business rule definitions, use cases and user stories or important user interface prototypes. Now mainly the developers, maintenance developers, users and user managers, etc. are the known beneficiaries of this document. In any development methodology, requirement is the primary goal to achieve before starting the development, as the desired product should meet the requirements mentioned in that document. However, in an agile way of development these requirements are preferred in form of living character (i.e. the customer). And the reason they give is that when the customer is available all the time, and the product is generated through a continuous sprint-wise evolution process then its better to have the customer involved, and that makes sense too, but I don't think the customer can be that much available for every project. So there must be an alternative too.

Rolf F. Katzenberger is a freelance practitioner who says

The "guideline" is best exemplified by the CCC structure of user stories (which are a popular agile way to capture requirements, but not the only one):

- you have the "Card": mostly, it carries a phrase like "As a <usertype>, I want to <activity involving the system> so that <business value>"; it's as short as that, because it's a mere reminder.
- you have the Confirmation: it specifies the acceptance criteria for the story; imagine writing an incredibly detailed script and agreement of how you will show and demonstrate that the implementation does what the story intended - then you select and delete everything but the heading level one texts and the remainder is the confirmation.
- And finally, you have the Communication part: a promise that people involved will take the time and keep talking to each other, to clarify what is required and to refine their shared understanding of it.

The Communication part is maybe what you are looking for. Agile teams do not follow a fixed set of rules on how (whether!) these requirements are to be documented in a written form. If the team is used to employ face-to-face communication only, that's it. The "guidance" is the shared understanding that evolves during the communication.

Sometimes, teams want more and they introduce additional, written documentation, as it seems fit - fine, as long as documentation serves the *global* optimization; plus, as long as the agreement still adheres to the values of the team and to the principles in the Agile Manifesto.

If you follow Scrum, for instance, some additional guidance is usually given by a product vision that keeps a Product Backlog focused, and by a Sprint Goal that does a similar thing for each sprint.

It's obvious that agile methods require *you* to think about and to define what type and amount of guidance the team needs. There is nothing mandated in the books. Being asked instead of ordered is a huge

As mentioned above by Rolf the most popular one is C.C.C structure user stories. (i.e. Cards, Communication, Confirmation), here in this comment we observe that communication is the major factor that will finally help you decide, what type and how detailed requirements document you will need. From communication, we mean both the communication between the developers and the customer as well as the very necessary communication of among the team members. The above comment shows the importance of the environment you are working in and its effects on the documentation part of your development.

One other idea that was very interesting and also very popular on the forum was from Wayne.

I realize you can't write a design document or a requirement's document for the project that will change with each sprint and never finish. Why not? Assuming there is a corporate need for the documentation, it is probably best to use the existing templates. For the requirement's document, create the shell document and paste the product backlog in order to serve as the requirements. Create this at the start of each sprint and bump the document rev level by 1. For the design document, a high level architecture diagram is usually valuable to outsiders and new comers to the team. It can serve as a basic index to start looking into the source code for details. This is also a good place to cover some of the system constraints that drive some of the design decisions. Detailed design, such as class hierarchies is usually not worth documenting - o to the source code for this. Turn over documentation is always valuable. Tell the help desk something new is coming. Let O and M knows how to tell the release is deployed correctly, is continuing to operating, and to determine it performance level." Another individual comment's that " Wayne's other points as to completing the documentation are largely correct, but can vary depending on the tools that the agile team is using (I-e automatically export from your story wall (software) to corporate requirement's documentation standard (Word doc or Req Pro, etc.). Alternatively, manually duplicate your physical story wall cards into the requirement's templates. Etc)." Having said that, being truly agile should reduce the quantity of documentation required because you address the risks that documentation mitigates through other means (I-e frequent in person conversation, daily stand-ups, showcases/product demo, retrospectives and most importantly working software). Therefore, only produce documentation for risks that can't be mitigated by other techniques (such as enduring support knowledge (turn over) as highlighted by Wayne).

Rolf F. Katzenberger replies in another statement,

```
it depends. It costs money to produce documentation artifacts. The business value obtained from that must outweigh the costs. "Outweigh" could mean, e.g.: * the customer can't wait to grab that documentation from your hands, and she's willing to pay for it. * the whole team, including the people writing those requirement documents, has found out these artifacts to improve their overall performance (globaloptimization). * there is a legal requirement for that kind of documentation (and the law just doesn't care about optimizing business value, just about annihilating it if you don't comply). So, all types of extremes are possible: I've seen teams that consider index cards with no more than short reminders as sufficient. And other teams that share extensive UX and UML documents, because they've arrived at that, through inspect and adapt.
```

Now from the above comment the factor that actually affects your documentation, 1st Rolf's point that distributed teams needs more documentation while more confined teams doesn't need that comprehensive documentation. For instance, if you have the customer in-house or building and is available all the time, then the requirement's documentation doesn't need to be that solid. However, if the customer is in another country or city, i.e. away from you, and you can't meet her/him in person but only once, then definitely you need to get solid requirements so that you don't need to worry about it in development. However, this thing is that in an agile way of development, the customer must be part of the continuous development process and must be available. Because it's a sprint wise development. And the customer's feedback and input are required at the end of each sprint and the beginning of another one. So the effect of the environment you are working in is prominent here.

Mikhail Seliverstov comment is

```
"@Rolf: great comment! I would also add that as a rule, the bigger/ distributed teams need more documentation, since they can't simply share the witeboard anymore."
```

So Mikhail's comment relates both to the size of the team as well as the environment.

Victor Hugo Certuche says in his comment that

The level of documentation is usually settled among the team members, and it is also dependent upon the complexity of the story. In most cases, in my experience, there is a great deal of detail in the acceptance criteria since it becomes the basis for TDD and contract acceptance. The rest of the documentation could be much lighter, for as long as it does not create ambiguity that later might cause disagreement on interpretation.

2nd point is of Hugo i.e. that it differs from a team to team depending the project orientation and project size. This too is one of the major factors, for example, smaller projects need less documentation while bigger projects need more detailed documentation. So Size of the project also is one of the factors that affects your documentation.

Brad Appleton says that

Requirement's documentation is vital to all projects, including Agile ones. The real issue is how loose or strict your definition of "documentation" is :-)

The real complaint of agile projects about documentation is one of the duplication/redundancy. I don't think any agilest would claim that knowledge of what to test+implement shouldn't be captured. It's a question of how much detail, ceremony and format. If information is captured in code or test-cases already, why does the same information need to exist anywhere else in some alternate format if it is at a comparable level of detail? The same is true for information naturally captured in any other format besides code/test too (like backlog items).

The real problem is how to capture the information once-and-only-once in a way that is readily "digestible" by any important stakeholder (not just developers). The typical knee-jerk response is to spend lots of effort to produce (and then maintain) a format that is more meaningful and useful to non-coding stakeholders of the project (and the more you have functional silos instead of cross-functional teams, the more of them there seem to be).

Naming matters too. If I ask whether or not requirement's documentation is important, I may get an a lot different response than if I ask whether or not end-user docs are important. And yet we should ask what is it that gets documented in a requirement's spec that doesn't eventually need to be part of documentation (online or otherwise) for end-users, administrators, or users of programming interfaces? Why would such end-user documentation have to be separate from requirements? Why can't one evolve into the other? Why can't examples in user-docs end-up as test-cases and why can't those be automated not just to execute them and see the results but also automated to the extent that I can capture the example/test-case in one place and have the other things automatically generated (so there is no duplication, at least not of the variety that requires syncing and multiple maintenance instead of just "rebuilding").

It seems to me that techniques like Agile backlog mgmt and acceptance TDD and BDD integrated with continuous automation for build, test, release/deploy to go a long way towards being able to auto generates

We can relate this comment to the experience of the developers, that how much detailed requirements they need to get the desired results.

This was the hot subject when we raised the question of requirements for software development in agile environment. Furthermore, we got a very detailed discussion on this one. We asked that,

- Do you think requirement's documents are documented properly?
- Do you observe use cases generated?
- Do you think user stories are developed with the help of users?
- User interfaces are depicted in requirements?

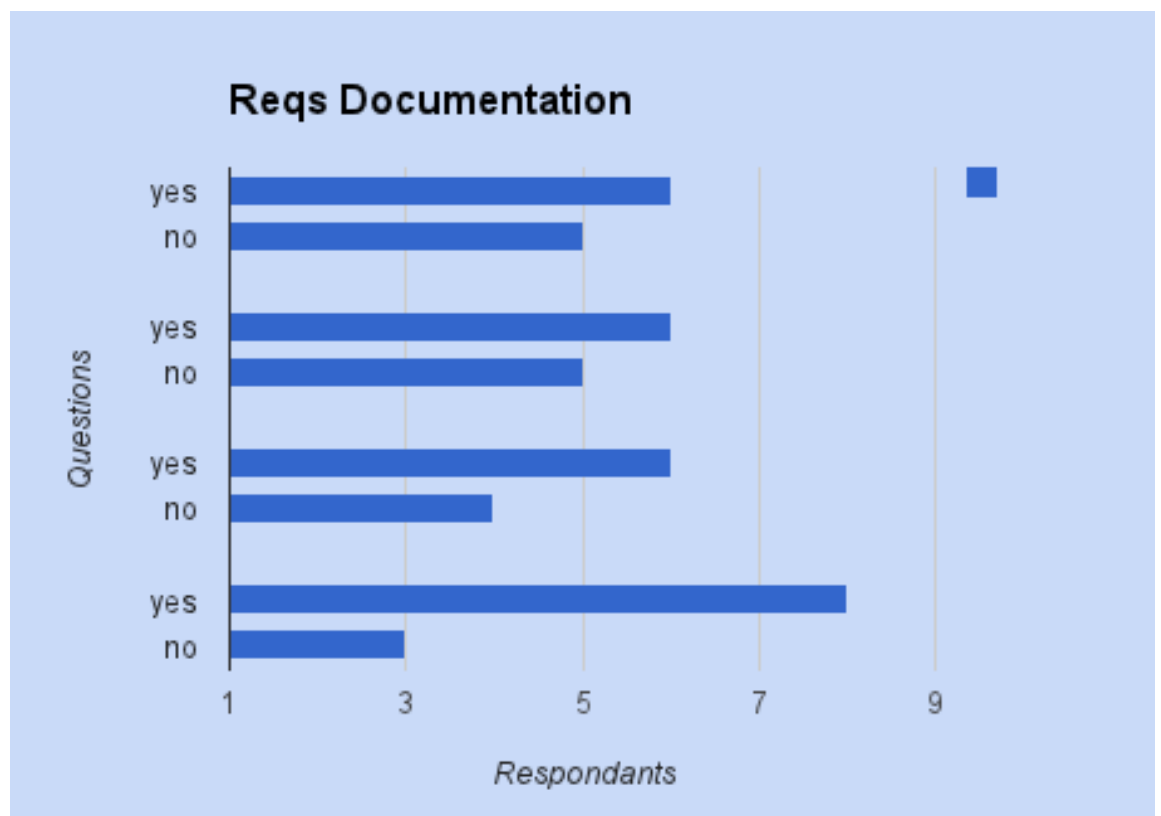


Figure 6.5: Requirements Documentation Graph

6.3.6 Support Documentation

Support documentation is basically training materials to help and support staff after the software development. This type of documentation may include support guide, user guide and manual guide, etc. these documentations evolves as the product updated so the support team also needs to be working closely with the developers in any story development or design meeting. It is the responsibility of the support engineer to follow the system as it evolves. User guide or support guide is a book type manual containing instructions on installing, using or troubleshooting software or hardware product. It can be very short like 10-20 pages, or it can be a full-length book of more than hundred pages. Following are the main components while designing user/support guide in a book form.

- Front and back cover
- Title pages
- Trademarks
- Edition notice
- Disclaimers
- Warranties
- License agreement
- Safety notices
- Preface
- Glossary
- Index
- Reader comment form [Biju \(2010\)](#)

Allen Holub talking about support documentation

Written documentation for an evolving Agile system is always stale.

The support organization should be working closely with the developers, and should participate in any story development or design meetings. As such, they should be intimately familiar to the system in its current state. Support engineers must be following the system as it evolves.

This is not something you can do with a bunch of passive people working for minimum wage in a large call center. Outsourcing your support to some external organization in Lubbock or Bangalore is usually dysfunctional because it's critical for user comments and problems to feed back into the story-development process. External organizations don't know how to do that.

So this comment refers to the environment's effect on your development of support document.

Brett Maytom doesn't think of support documentation that high and says

If the document is going to be **really** used by support, then by all means create it. I find that most documents don't get read at all. I have seen projects with gigs of documents that took thousands of man-hours to produce, and they were only read by the author and proof readers.

If you find a document will **really** add value, then do it; but don't create documents for the sake of it and because it "might" be useful. Eliminate waste.

but we can say that customer requirement of the support document is of high importance in his opinion. Which definitely is of very high importance, not only what they want to have in documentation but also what you as a developer think they would require in support.

Steve Blais narrates a scenario he faced himself

Interesting. Just this past week we got a request from a group of users for some written documentation explaining what the new features were going to do so that they could be prepared. One of the users in the meeting said that they were expanding (don't know what that means) and needed written documentation to give to the contractors and new employees. Coincidentally, in a subsequent meeting on a different aspect of the internal system, a couple of trainers asked for written documentation as well. So far, the scrum teams have resisted such requests and have the backing of the software-development manager. While it keeps scrum 'pure' - no documentation that they won't read (as referenced above) and keeps developers developing, product owners owning product backlogs and scrum masters mastering, there is no one to respond to the requests. Does the company just tell these stakeholders they are on their own? I am not sure we as a society are so computer sophisticated that everyone will naturally and instinctively figure out how to use a given system. At least not yet.

This is not only in this one case, but customers and users are facing and complaining this problem on a quite large scale now. Support documents are the most important documents that accompany the product, and can extend the life time of the product. So we can also discuss the Types and level of users the developers are dealing with, Because these users are the one supposed to deal with the support documents. The more the level of users, and the more diverse the support documents would be

Michael Nir expresses his thoughts with very strong points and says

NEWS Flash - Boeing 747-400 crash-landed - main SW Navigational system showed Mt. nearby - Black Box reveals the System had no user manual as the Agile SW developers said - any moron can figure what to do when the red lights blink...

Steve - Thanks for all these thoughts - RTFM - had to look that up - I understand why no one bothered to add the 4 word description...

I have to admit I am getting a bit tired of some comments here and elsewhere regarding Agile purity and some fanatic views - so let me use a metaphor to get my point across. I have a client in the pharmaceutical industry - they just went Agile on some drug development - I think that now as they are using Agile they should stop handing out that medical documentation for Advil - you know the dosage and stuff like that - if you are stupid enough not to know the correct dosage - then the world is better off without you. And they should also stop writing down side effects - if you have them, you'll know that anyway.

Or better yet, the software developers for the main aircraft navigation unit in Boeing just went Agile - they delivered the last upgrade to the 747-400 SW and told the pilots that they are better off just calling them if the system starts issuing red lights - since all this documentation is an overhead, and the updated SW is due within four months anyway, and nothing stays the same and life is a big river of change...Hey I buy into that.... Would you want to fly in an Agile no documentation main GPS system on Boeing your next flight?

Let's climb down from this tree and be somewhat more compromising - we need documentation for users. We need it smart, and we need it useful - probably not a 396 page documents rather an on demand user friendly manual.

It can't agree more, this is a perfect phrasing of situations like these. This signifies the importance of the Support documents. And shows that you need to think of what the customer or user requires to get to help themselves.

6.3.7 System Documentation

System documentation is to provide an overview of the system and actually to help the team members understand the system. This document is specifically for the developers and maintenance developers. The information provided by this document includes technical architecture showing the technical aspects of the system, like how its working and its vital components, etc. It may also include the business architecture of the system, i.e. to provide an idea to the developers what the system is going to produce and how is it going to produce it. The technical and business architecture shows that the system documentation should at least include high level requirements for the system. And so the detailed architecture and design models (if there are) should also be presented here in this document. System documentation means to provide accurate information to the customers that allows them to effectively use and maintain the product. This type of documentation should be written that a person with at least knowledge about that product can understand easily. System documentation usually provides an overview of the system that the people can understand the system. It gives more benefit to the people who are new in the project so those people can understand more things from system documentation [de Souza et al. \(2005\)](#). Some desirable characteristics of good system documentation are given below.

- Created for intended audience: System documentation should be created for specific audience, and it will provide a sufficient technical details. Mostly such type of documentation will be used by the system administrators and other technical people as an importance reference.
- Specific: This type of documentation will provide specific implementation of the give system rather than provide general information.
- Relationship with another documentation: Such type of documentations is self-contained document. However, most of the time it will be a component of wider collection of documentation, and it is used as a reference to other documents.
- Up to date: System documentation will be up to date, and it does not mean that it will be the most-recent document. Document will be updated as there any changes occurs within the system.
- Sufficiently comprehensive: System documentation will be comprehensive that it can portray and present its purpose clearly.
- Accessible: Documentation will be located in such a place that everyone can access easily. Format of the documentation will be in a universal standard format so that it does not create a compatibility problem [Kajko-Mattsson \(2005\)](#).

Now we will discuss some of the statements given by some very experienced practitioners, try to relate them to the components that actually affect your over-all documentation. Like the way Allen Holub says that

```
No, the system should be self-documenting. Out of
date documentation (and most documentation is out
of date the day after it's written) is worse than no
documentation at all. Writing truly self-documenting
code is difficult. However, and requires real upper
management support, without which people will think
that they "don't have the time" to do it right.
Having said that, some sort of lightweight, easily
up-datable, architectural view (a stripped-down UML
class diagram on a whiteboard, or a set of CRC cards
tacked up on the wall, for example) is helpful. If
you don't have the discipline to keep that diagram or
those cards fresh, however, then skip it.
```

Allen is sounds to be more focusing on the code, and believes self-documentation of the system. Which he describes as either a stripped down UML class diagram on a whiteboard, or a set of CRC cards tacked up on the wall. Or you can adopt any other similar kind of approach, because the purpose of this practice is just to remind yourself what you are doing and to have a clear vision of what's been done and what's still to be done. We can relate here the tool's segment as the tool that you use for your documentation is also of key importance, and decides how your documentation is going to look like.

Steve Blais is another very experience individual in the area, and he thinks little different than Allen. We picked the contrasting comments to show that in the market or in practice, you will see all aspects and different versions of a single practice. Steve has a very good point and is expressing that how sometimes only well documented code or no other documentation than the code can get you in trouble. He says

Allen - It is nice to have some kind of system architecture, network topology, data model or software overview diagram, UML or not when you come into an existing system and are told that you need to replace a major section of it. If, as Allen says, the system is self-documenting, then the documentation should be in the code. (Unless there is another way, a system documents itself that I am not aware of). I don't think we want to read comments in several hundred programs to try to understand the big picture of what we have to do and, more importantly, learn what areas we should not be changing or impacting. I'm not sure of what kind of "management support" you are referring to, Allen. Most upper-level managers have no concept of what code is much less what should go into it. About the best you can get is some lip service policy that "all code should be commented." Management certainly is not going to check it out. As for trying to salvage the big picture of a system from a set of cards or half-erased and smeared white boards, well... And, yes, in Agile we are supposed to talk rather than document, but how much time can the individual developers give to us when they are on sprint deadline and working in the details? Each developer and each team has a product backlog, and potentially a different product owner, all parts of the same overall system. If we are under a deadline to get our new replacement system defined, visiting every developer working on the current system seems onerous at best, and we still have to piece together all the anecdotal information. Are you sure you should not practice system documentation in agile?

Steve is presenting how the environment and the situations you are working in can affect your documentation. In different environments, your requirements of the system documentations are different.

Jayne Hamilton is referring to a very important point, and a problem that many developers and teams face during their development. That is that sometimes the management or the customer may demand certain requirements. That could either be more detailed documentation and could also be that they force you to skip the documentation parts, and the time you need to do your

documentation. She says

```
Sadly, this "we don't have time to make the code
self-documenting or re-factor. What has been built
in an unclear way into clean code" from management can
lead to major impediments in the future. A stitch in
time saves 9. I have even met one manager who thought
it was not useful to write unit tests. In these
cases, I refer to Uncle Bob's (Robert C Martin's)
nice Devovx presentation who urges developers to "do
no harm". If a bug appears in a crucial production
system and code has to be researched, clean readable
code makes it easier to see what is going on and
the issues are spotted faster. If a company puts
developers under so much time pressure that clean
code principles and quality go out of the window,
it is really shooting itself in the foot. As for
documentation, a tool like Maven site can be very
handy... It's included in the code base and not as
heavy as some documentation tools. However, we do
need some system documentation somewhere. It's vital
for knowledge transfer processes for the future.
```

Customer requirement is the segment that we can relate this comment to. This does not mean that customer's demands are always counterproductive. Sometimes they do need extra documents or more detailed documents to either help start the fresh users or to help them in future maintenance.

6.3.8 Operation Documentation

This documentation typically includes your system's dependencies that it is involved with. It actually shows the nature of its interaction with other systems, databases or files, etc. It could also be like a list of contact points for your system and how to reach and interact through those points. Availability/reliability requirements for your system could be a very important part of this document. Now it sounds too much to some practitioners, especially in agile environment. However, it depends upon the type of product you are working on. In larger projects or when you are working on a system/product that has frequent or permanent interaction with other systems. Then this document becomes more and more important. A system that has dependencies on other systems or sources, then you cannot rely on that system until you define those dependencies and make sure they're available and reliable. This is the document that handles it. Below are some of the responses we received when we raised a question about the importance and practice of operation

documentation during the development process. Just to remind that these questions raised are on an Agile group specifically, i.e. a couple of Agile specific practicing groups. Scott W. Ambler says who is a Senior Consulting Partner at Scott Ambler + Associates and is a very well-known name on Agile concepts.

"A few thoughts:

1. The value of documentation, of a deliverable in general, is determined by the customer of that deliverable.
2. Disciplined agilists recognize that they should be producing potentially consumable solutions on a regular basis, not just potentially shippable software. That solution potentially includes software, deliverable documentation (of which operation's docs might be included), changes to hardware, changes to the business process, and even changes to the organization structure of the people using the solution.
3. Disciplined agilists also recognize that operation's people are an important class of stakeholder for them, an aspect of Disciplined Agile Delivery's inclusion of DevOps concepts throughout the framework.
4. You might find my thoughts on agile documentation at <http://www.agilemodeling.com/essays/agileDocumentation.htm> will provide some insight.

The short answer is that it depends upon the context of your situation. For some team's operations documents are very important, for others not very important. And anything in between."

Scott's opinion is that the importance of operation's documentation varies from a group to group, and the 'context of situation' might be the type of product the team is working on. For some products, you really need this document and is of a very high importance while for other products, it could be relatively less important. However, as I it's been said at the beginning that it also depends a lot on the dependencies of your system or product on other resources and the dependence of the consumers on your product. So we can relate his comment to two components the 'type of product' and then, to some extent, to 'team experience'. Joseph Bechenbach says who

is another Software Developer at HUNTER Technical Resources, LLC

"Depends on the context of the team and the project. I'm often in Agile teams, which operate in production the software we write ("devOps"), with non-team peers also responsible for making sure the services remain available. What those non-team peers need to know, this need for "operation ((of the software) documentation spawns deliverables for the team to provide. Similarly, the operation of the team itself needs to be known. Some can be taught through normal pairing and training up new folks in the "team culture." Other aspects might need documenting, such as how the continuous-integration server works and how to upgrade it. To me, that "operation ((of the team) documentation also spawns deliverables for the team to provide, though primarily for the team itself. Occasionally, a manager or a peer from another team wants a look at them, to confirm business continuity or to figure out process improvements. It's good to have something crystallized into writing for ease of reference then."

Joseph thinks that practicing operation's documentation depends upon the type of project the team is doing and depends also on the context of the team. So in this comment we can relate to the component of 'type of product'. Scott Nelson who is a Director of Enterprise Portal Architecture gives his opinion in the following way;

Adding to what Joseph said, it also depends on whether the development team also does Operations and Maintenance. The less the development team is involved in OM, the more important the Operations Documentation is as well the higher standard the documentation has to fill. Thinking back on a few applications I have worked with, it would also depend upon the stability of the contracts involved. For example, in some government agencies, the contracts can completely replace the development team in a very short time frame. In such cases, even with the developers doing most of the maintenance the documentation is critical if there is a chance, they will suddenly need to turn it all over to a new vendor."

Scott Nelson's comment makes a very solid argument; he says that the amount of your operation's documentation depends upon the involvement of the team in the Operation Maintenance. If the same team will be doing the OM, then operation's documentation does have to be that much detailed, but if the developing team is not going to be doing the OM, then you need to have a detailed operation's documentation to help solve future problems. Brett Maytom works as a Principal Consultant at Readify, and is a Professional Scrum Trainer, who

Mikhail Seliverstov comment is

"What value does the document provide and the real question is whether the document **really** gets used as well as the longevity of the document. Thus if the document is just for handover to one operations team and does not get used post the initial deployment, then its worthless. However, if the operation document gets given to thousands of customer operation teams, and will be used by many people as they do not have access to the development team, then I would say its pretty important."

Brett's opinion means that it depends upon the type of product and the customer requirements probably. Type of product has to be highlighted previously too, but customer requirements also play a very vital role in deciding which type of operation's documentation should be provided along with the product and also how detailed that documentation should be.

6.3.9 Contract Model

A document describing the technical interface to a system or portion of a system. It's a very common known subject and is not much different than it's practiced in other developing methodologies. As its part of documentation which makes it our ninth component. Following are few comments that which clarifies that contract model is implemented almost the same way is it is in any other methodology. Victor Hugo Certuche is a Manager Business Consulting at Sapien, and does not see practicing contract model to be more different in Agile than that in other developing methodologies.

"Contract Model is just a form of specification and can and has been used with any methodology. Not sure why you see it different with Agile. Perhaps you can elaborate a bit."

Paul Oldfield is one of very experienced developer and currently works as a team member at Youmanage HR Ltd. Paul says;

"It's an optional approach. "Design by Contract," which takes the contract model well into the design realm (if I'm not mistaken over your meaning), is not commonly used from what I've seen, but there is nothing to say we shouldn't use it. Commonly used is TDD (Test-Driven Development) that has been several of the same good effects as DbC."

Chris Olufson works at Integration, SOA Architect, Service Oriented Architecture, Integration and Cloud.

"Works for me. In my world of SOA, I can deliver software service contracts (WSDL + documentation + samples), and the consumer of the software service can commence their work, whilst I complete the implementation of the software service. The delivery is in two steps.

1. contract
2. software (realizing the contract)

6.4 Interpretation of Data

So even if we admit that working software is of more importance in an agile way of working than that of large detailed documentation, the importance of vital segments of documents cannot be neglected. Now which components could be included and which one excluded totally depends on

the developers and the customer. By saying that it depends on the developers is because practically its decided by the developers that which document is important and should be developed and which one is not important. While the customer has the privilege to demand for certain documents that she/he requires. Figure 6.6 presents the factors which affect documentation. But the requirement and importance of different document's changes from a project to project, so it's more dependent upon the type of product being developed, the size of the project, the environment the developers are working in, the experience of the team, customer requirements, the number of users and the level of users.

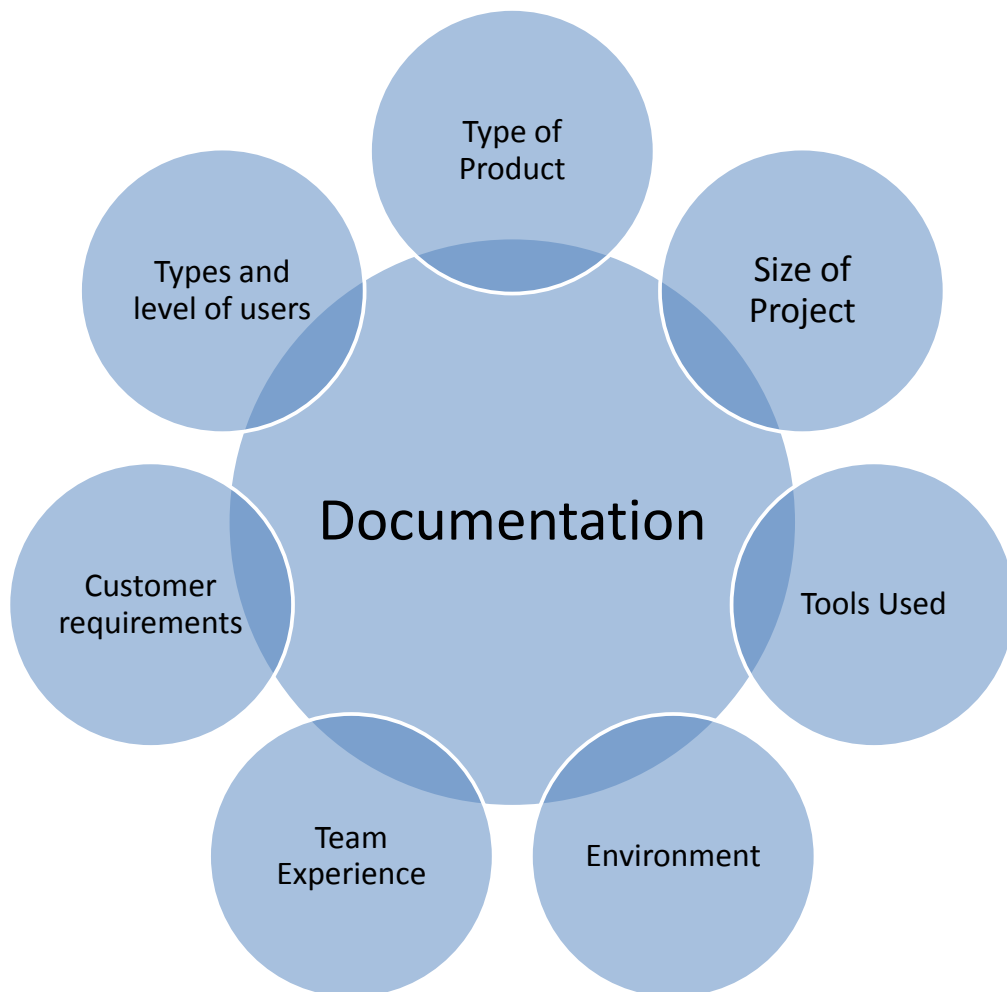


Figure 6.6: Factors which affect documentation

6.4.1 Type of Product

Product type is one very important factor that helps you decide that how many different types of documents are to be developed and how detailed the documents should be. Even there could also

be documents for which you don't need any operational or user documents. As mentioned operational documents because you always need process documents, the actual reason of the generation of process documents is that all the theoretical work done during the process of development. That means you have to generate these documents, whether you need it or not. For example, if you consider requirement document, if you are developing a tutoring software for students to be used in a school to learn a language. For that you need a detailed requirement document to know what sort of a product exactly the customer is looking forward to and how can you make it much better. You have to meet the expectations of the customer. On the other hand, if you have to develop a code to help machines run, that just operates automatically and does not need configuration or human interaction. Then here you don't need as much documentation as you needed for the previous type of product. So the type of product does affect your documentation.

6.4.2 Size of the Project

Size also has great importance, the bigger the project and greater and more detailed the documentation would be. There are many factors in larger projects that affect your documentation too. For example, if the project is big, then you also need a big team and more crew members. The increase in the team members also increases the amount of communications among the team. Furthermore, due to different minds working on different components, you need to have a proper authentication and testing system. Because you don't really know whether you are getting the right thing. That increases your process documentation. Furthermore, with the size the budget and the scheduled time to implement the project also increases. And for both these things, the budget and the schedule, you need to keep a proper record of everything and generate reports on a regular basis. So the above discussion proves that larger projects have a great impact upon the size of your process documentation. Aside from that it also affects your operational documentation. Because the more complicated the product and the more help you need to not only get it installed but get started working with it and maintain the product.

6.4.3 The environment

The environment means both the environment in which the product is developed and the environment in which the product is meant to be operated. Environment always has a great impact on your work. The efficiency of your work or team work is directly proportional to the environment your team is working in. That also means the documentation. For example, if your team is sitting under one roof, then the coordination and communication is much better and so is the understanding of the things each member is working on. In person meeting can never be replaced by anything, and that's what the agile approach has always talked about. This decreases the unnecessary documentation to a great extent and rest important documentation is much precise and to the point. So we have to understand that agile does not mean that you have to avoid documentation. the point is that you to increase your coordination and communication to such a level that the unnecessary

documentation is eliminated by itself. Let's say if your team is scattered around the world or sitting in their houses and are communicating through reports and emails. Then it definitely has an impact upon the efficiency of the team work too. So the environment is very much important in agile methodology.

6.4.4 Experience of the Team

Experience is a very vital feature for any project. If your team is experienced and has done the similar kind of job in the past, then things become much easier. If your team is fresh and has no previous experience of a similar kind of work, then person or group has to start from scratch and has not only to understand the phase of work but also to get used to and fit into the environment as soon as possible. Inexperienced team could result in disaster. It does not only affect your documentation, especially process documentation but also could have a great impact on the overall development. Sometimes the team works on a project that is new for them, means they haven't been similar kind of project before. So in that situation also the team has to document every single part of the development process. Because it's the first-time experience and there need to be a proper record that could help understands the weak and the strong points of the project implementation. However, as the team works on more and more of these projects, the unnecessary documentation eliminates by itself. So it means that inexperience team would always have to go through more documentation as compared to much experienced team.

6.4.5 Customer Requirements

Customer requirement also changes the importance of documentation. And note that is not a customer requirement document we are talking about we are talking about the overall documentation which includes requirement's document too, but the emphasis is more on user and operational documentation. Because these are the documents that are related more to the customer. And in some cases if you are working for a company and they require you to provide certain documentation along with the work done. In this case, the company is your customer, and you have to meet their requirements. This requirement of the documentation could be either to understand the development process, that how the product has been developed. And it also could be for the maintenance purposes because some systems or products are so much complicated, and you can't be sure about the availability of the developer so in that situation if you have the necessary documentation it could help a lot in fixing the problem.

6.4.6 Type and Level of Users

The more the number of types of users the more detailed the documentation would be required. Mostly, the users are divided into groups and then the required information for the specified groups is provided with the help of different types of documents. These documents may change from a project to project depending upon the type of product the team is working on and the requirement of the user group. Because the information required to the user may change from one project

product to another project product. Similarly, these documents may also change from one user group to another user group. In this case we can term, it as level of users. There is no standard defined level of users, but generally, we would observe users like administrators, maintenance technicians and end users, etc. Now these groups can even be further classified depending on their interaction with the s/w or system, and so the information required to them are provided in the same perspective. You cannot provide maintenance information to the end users; otherwise, it could result deadly for the system. Similarly, the end user information does not provide enough help to the maintenance technician. So all these documents are supposed to be made available if you are dealing with the respective user groups. And so the more the user types and levels the more the documentation would be required.

6.4.7 Tools Used

In agile methodologies different types of tools can be used for different forms of projects to develop well structured documentation. Before choosing a specific tool to develop documentation, it is important what the purpose of documentation is and who is the reader and audience. During development tracking story status with team members is different to deliver a hand off document to product owner or client. Whatever the reason, to develop documentation one will need a tool that can help him along the way. Different tools can be used by different people in different projects. Some of them are the following.

- Ms Word (and other word processors)
- Javadoc and similar tools (Doxygen, Doc++)
- Text Editors
- Version one
- Jira
- Contour
- Excel

Documents can also be stored in form of plain text, HTML or XML file and maintained by the programmers. Other tools and technology found by the technical writers are ArgoUML, Visio, FrameMaker, Author-IT, digital cameras, whiteboards, JUnit and XML editors. [Forward and Lethbridge \(2002\)](#)

6.5 Standards

During the past six years, group of International Organization for Standardization (ISO) collaboration with International Electro technical Commission (IEC) produced international standards for

software user documentation. They produced five standards, among those five standards four of these are directed to various audiences while the fifth is devoted to produce documentation in an agile software-development environment. Five standards are given below [Hayhoe \(2012\)](#).

- System and software engineering requirements for managers of user documentation, ISO/IEC/IEEE standard 26511, 2012
- System and software engineering requirements for acquirers and suppliers of user documentation, ISO/IEC/IEEE standard 26512, 2011
- System and software engineering requirements for testers and reviewers of user documentation, ISO/IEC/IEEE standard 26513, 2009
- System and software engineering requirements for designers and developers of user documentation, ISO/IEC/IEEE standard, 26514, 2008
- ISO/IEC/IEEE 26515:2012(E) System and software engineering development user documentation in an agile environment.

The first edition of a standard document regarding documentation of software development in an agile environment was produced on 1st of December 2011. Later then another corrected version was issued on 15th of March 2012. The standard model ISO/IEC/IEEE 26515:2012(E) (Titled Systems and Software Engineering- Developing user documentation in an Agile Environment) is the latest standard. Now this standard has been referred as an assistance to the users of ISO/IEC 2651 n family of international standards along ISO/IEC 15288:2008 and ISO/IEC 12207:2008, but the last two are not subject to documentation but the development process cycle of system and software (Standard ISO/IEC/IEEE 26515:2012(E)).

Chapter 7

Discussion and Conclusion

This last chapter summarizes the whole thesis, it discusses that how documentation should be produced, what is our contribution to previous research work and weaknesses of our research work. In the last segment, conclusion of our thesis work..

=====

7.1 Discussion

For each segment that has been presented to ask questions about, a very diverse set of experiences and opinions has been shared by different practitioners. None of the practitioners denied the importance of documentation, so based on data collected in this study documentation in an agile way of development is as important as it is in any other software development methodology. But the importance of different segments that has been used to collect opinions varies from person to person and sometimes depends on your project orientation. Based on all possible problems that a developer faces during their development, Seven major factors have been identified that actually defines how you should be developing your documentation during your development of the product. Now for the understanding of the readers every single opinion or comment that was received has been related to the respective factor. That 7 factors diagram is our suggested solution to identify the type of documents(Internal documents) you will be needed for a particular product or to identify the problems that the development teams faces. It becomes very clear to any developer or a team to identify the problems that they will be facing during their development, which varies among practitioners and from one project to another. The direct impact of these factors can be greatly reduced by having complete knowledge and understanding of these factors.

1. What are our answers to the research questions?

Q 1. How should documentation be produced and used in agile development?

It is clear from one of the recently done survey, that we presented in the thesis that the developers find internal documentation to be quite effective, yes face to face communication

is more effective, but in some cases you do need internal formal documentation for effective results. Agile methodology never says that you must always be under one roof as it is not possible in current global environment and everyone want to use skills and experiences from across the globe. Therefore, it is very important to keep the doors of different experiences open in the agile way of development, and let the evolution of this methodology continue. We would suggest that before starting your project you must identify the challenges that you possibly would be facing. That is what we have tried to present in this thesis. Documentation has its natural division of process and product documentation. We have given three factors related to product documentation, that affect your product documentation. i.e. Type of the product, Types and level of users and Customer requirements. These three factors helps decide what you need to develop to support your product and will be helpful for the user. These three factors will probably address all the problems that rises to both the developers of these documents as well as the problems faced by the users. Type of the product defines to some extent, what sort of documents must accompany the product. There is a vast variety of products that has completely different function, for example, documents required along with an application software for casual users can never be the same as the documents required for software to configure a hardware. Types and level of users also have to be kept in mind while developing a product. Type shows how many different types of users will use the product and level shows how much knowledge each of those users already has. This information related to users will help the document writer to provide adequate information to the user, regarding the product. Customer requirements are usually the direct or indirect demands of the customer. The comprehensiveness of these documents depends on the previous factor, i.e. level of users. Then for process documentation there are three other factors, the size of the project, Team experience and the environment. These are the factors that affect extensively your documentation during the process of development. Size of the project defines the structure and amount of documentation. The bigger the size of the project the more the amount of documentation and the bigger the communication structure would be. For example, a team of five or ten can have a daily scrum meeting with no complications, and these meetings could be very favorably around the table under a one roof. However, if your team consists of 50 members with a different level of experiences and very distant from each other, then it becomes very important to come up with a structural communication and coordination system, so size does matter. The second factor that affects your process documentation is the team experience. It is very important that if not the whole team at least some members should have adequate experience of the same type of projects. Experience of both the project and the documentation. Agile teams usually evolve and grow their skills with experience. A more experienced team would not waste time on unimportant matters and would be more efficient and focused as compare to an inexperienced team. The third factor of process documentation is the environment. Environment could multiply your efficiency if managed properly. Now there is a seventh factor, it is the tools that you used in the development of your documentation. Different tools generate different types of informa-

tion, or the same information in a totally different format or document. The tools used for the formal communication also affect on your process documentation. So this is the factor that actually affects both your product and process documentation. Hope the knowledge and study of these factors will help the developers that how documentation in the agile way of development should be conducted, and what sort of challenges could a team face during the development process.

Q 2. What is the rationale (motivation) to produce and use documentation in agile development?

We have two different types of survey studies in this thesis, one which we did with the help of a social network. In this survey, we got opinions and discussions from very experienced practitioners and none of them disagreed with the fact that documentation is not only an important part of the agile methodology but any development methodology. Now chapter 5 (i.e. Literature discussion) answers this question in a manner of detailed discussion as well as some results of a recently done survey. This is a very relative survey which we could not help not to include in the thesis. However, before we discuss the questions raised and the interesting results obtained during this survey, let us review these opinions of some of the very experienced practitioners and agile researchers. These are considered to be the pioneers of agile. For example, Scott Ambler says that documentation becomes out of date and should be updated only “when it hurts”. Sometimes it is necessary in order to retain critical information over time. Ambler also suggests two primary reasons for documentation, namely that we should model (or document) to communicate, or model to understand. He says that the documents to be produced to support the thinking associated with each stage during the project. In another comment, Barry Boehm says that a documented project makes it easier for an outside expert to diagnose problems. Barry Boehm’s is raising a very important aspect of why we need quality documentation. This is one of many uses of an up-to-date documentation. Ambler’s point of updating of documentation is required otherwise the document loses its effectiveness. Bil Kleb says that “with agile methods, documentation is assigned a cost, and its extent is determined by the customer (excepting internal documentation).” Now it is natural that documents demanded by the customer are compulsory to develop, and as said by Bil Kleb. He is excluding the internal documentation. Internal documentation is actually the part that agile tries to minimize and make it more efficient. Because it has no direct effects on the user or customer, but it definitely has direct effects on the product so it is very important that only unimportant practices should be eliminated. Now the rationale behind the documentation of a project is evident in the results received to the questions raised in a survey that we discussed above in chapter 5. Just to remind that it is the internal documentation that actually is considered to be the disputed one in agile methodology, otherwise the user support documents are compulsory and have to be developed. Survey is only about the internal documentation. It’s called “An empirical study of internal documentation in agile software-development teams”. Now if we study this survey,

when they are asked that how effective do you find internal documentation? So a vast majority finds it effective, or is either neutral. The effectiveness of this documentation itself is a motivation for the practitioners. Now the second question they were asked was that how do you feel about documentation at work? Now keeping in mind that the results provided shows that almost half of the developers think of their documentation to be too little. This means they want the documentation to be rather more detailed, and they think of the current documentation as insufficient. These results show how the agile practitioners feel about the agile approach towards internal documentation. The third question asked was that how important do you consider documentation for your project? We would expect that agile practitioners would rate internal documentation as not so important but the results for these questions are remarkable. As majority of participants defines documentation to be either important or very important. The results show how internal documentation is considered around the world among practitioners.

2. What are the contributions of our work in relation to previous research?

Most of the researches done related to the area of agile methodology are more focused on the working of the methodology, looks like researchers were not paying that much attention to the area of documentation. The reason of this negligence could possibly be the already present perception that documentation is fading away in an agile way of working. This was one of the main reasons of our interest to find out why its is this perception that if documentation becomes less important when we talk of an agile way of development. On the contrary, of that when we study any relative study that covers documentation in agile methodology, the product documents are as important as in any other software-development methodology. However internal documentation could shrink as much as possible as the methodology is designed to be more focused on daily progress reports in daily scrum meetings. And the in person daily basis meeting eliminates most of the paper work required for communication and coordination, which is understandable. Now due to the agile nature of this methodology it's impossible to have already defined set of documents that should be prepared by the agile developers. So we selected the nine segments that could possibly cover all documents. These nine segments are then been used to collect data, what the practitioners think about each of these segments. It was not only the collection of data but there has been a detailed discussion on each of those segments. All the practitioners identified the problems that could be faced during the development of documentation. Based on all the problems identified we defined seven factors that affect your documentation during development. Now each of the problems identified during the discussion has been related to the respective factor in Chapter 6. We have given a review of the main purpose of documentation from the literature. To know the essence of documenting a process, we collected and presented opinions of some very important researchers which are considered to be the pioneers of agile methodology and identified problems with help of practitioners. Then we defined seven factors that actually affect your documentation, which we consider to be the

outcome of this thesis and a suggested solution for identifying and handling your problems that you face during the development of documentation.

3. What are the weaknesses of our approach, that the reader should have in mind when interpreting our results?

Well, we have taken our data from two means, first we did our own survey and the reason behind this study was to get opinions and to identify problems they experienced during their practices. This was not a number survey, but was actually more focused on their comments and opinions. That's why some of the comments are included in the thesis. Then there is a survey called "An empirical study of internal documentation in agile s/w development teams". This survey was giving us exactly what we needed to ask the practitioners and conclude our results. So we got permission from the original writers of that work. We have picked the questions that they have already raised and presented here, along that we have used the graphs that they generated from their data believing it to be correct.

7.2 Conclusion

Documentation is not bad but bad documentation is terrible. Agile methodology values face to face communication over formal processes. Agile documentation always focuses on light weight, relevant, up-to-date documentation while in traditional documentation. Agile never forbids documentation but it expects from the practitioners to just go for the important one. Agile always focus that documentation should be light because the system is constantly changing, it is easy to make changes in the light documents. This methodology gives more attention to working software and by working software it means that most of the documentation should be inside of the code. Agile is a people-oriented methodology and when there is more interaction between developer and customer, more chances of less documentation. On the other hand, too much documentation could also be hard to maintain and update. We collected data through linkedin surveys where we asked different sets of questions regarding documentations, and the answers were given by different agile practitioners and opinion makers. The set of documentation used to collect data are, User documents, Design decisions, Vision statement, Project overview, Requirements document, Support document, System documents, etc. All the practitioners and opinion makers were agreed on one common point that documentation should be concise, correct, well-defined and up to date. Our conclusion from this work was that documentation depends upon the needs of the customer and the developer. There can't be a defined set of documents that must be produced along with every product during the development in an agile way of development. Every team decides should decide their own set of documents according to their needs and environment. So the factors we introduced were to help the developers and practitioners decide what they lack or what they need to overcome the problems they are facing. To further clarify there need to be a step-wise process or procedure to produce documents. Now the process of documentation in an agile development process should perform the following important operations or activities.

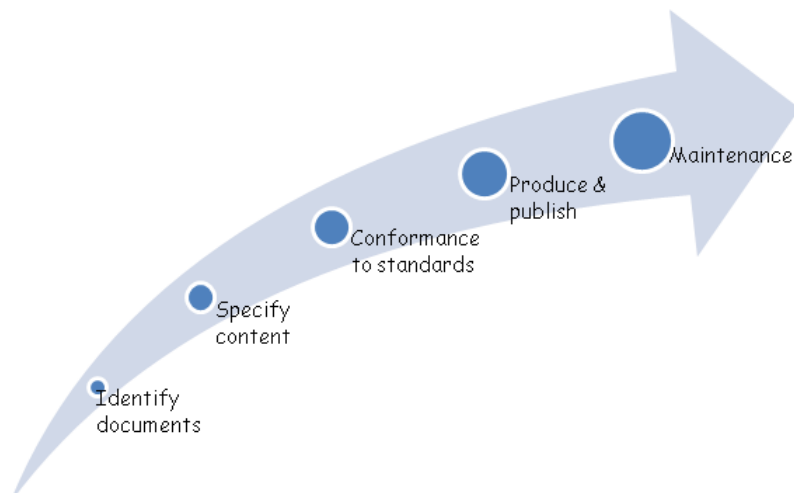


Figure 7.1: Operations to produce documents

1. Identify documents to be produced during the project or process.
2. To specify the content of a document then the purpose of the document and then the plan and schedule its development and production.
3. Identification and conformance to the adopted standard.
4. Produce and publish the documents in accordance with nominated plans.
5. Maintain the documents in accordance with specified criteria.

Performing the above process to produce your documents, keeping in mind and having the knowledge of all the seven factors that will affect your documents during this development process. This is our conclusion which we hope will work!

Appendix A

Questionnaire

This appendix presents the Questionnaire based on the Nine components of documentation defined in the thesis.

A.1 User Documents

1. Which user documents do you practice?

- (a) Reference Manual
- (b) User Guide
- (c) Support Guide
- (d) Training Material

A.2 Design Decision

- Do you practically document design decisions?
 - Yes
 - No
- How often do you practice it?
 - Frequently
 - Occasionally
 - Never Practiced
- Do you document system design and architectural designs in development?
 - Yes
 - No

- Do you think it's important in agile environment?
 - Yes
 - No

A.3 Vision Statement

- Have you observed such precise statements with cost estimates and time schedules from executives?
 - Yes
 - No
- what do you think what are it's uses and why it's required?

Type your text here.....

- How flexible are these schedules and budgets?
 - Flexible
 - Not Flexible
 - More Flexible
- How often these schedules are met?
 - Always Meet
 - Sometimes Meet
 - Never Meet

A.4 Project Overview

- Do you practice or see project overview summary practice?
 - Yes
 - No

- What exactly the project overview consist that you implement?

Type your text here.....

- How detailed project overview summary should be?

Type your text here.....

- What is the main purpose or usage of this document?

Type your text here.....

A.5 Requirements Documents

- Do you see use cases generated during the development process?
 - Yes
 - No
- Do you think requirements are documented properly and timely?
 - Yes
 - No
- Do you think user stories are developed with the help of different users?
 - Yes
 - No
- Do you observe possible user interfaces are depicted in the requirement phase before designing start?
 - Yes
 - No

A.6 Support Documents

- What types of support document you exactly generate?

Type your text here.....

- What are the specific uses of each of those documents?

Type your text here.....

A.7 System Documentation

- How important system documentation is in agile environment?
 - More Important
 - Less Important

– Not Important

- Which System document exactly you practice in your projects?

Type your text here.....

- Can you elaborate the parts that you implement during system development?

Type your text here.....

- Do you practice or observe structure designs and architectures practiced in agile environment?

– Yes

– No

- Are those design models or documents included as part of system documentation too?

– Yes

– No

A.8 Operations Documentations

- Do you practice or have seen operations documentation practiced in your company?

– Yes

– No

- Can you elaborate how it its practiced?

Type your text here.....

- How exactly do you document the dependencies of a system?

Type your text here.....

- Which document do you use to cover the availability/reliability requirements?

Type your text here.....

- Do you think availability/reliability requirements summary is important?

– Yes

– No

- How precisely do you practice it?

Type your text here.....

A.9 Contract Model

- Do you practice Contract model?

- Yes

- No

- What is your understanding of contract model?

Type your text here.....

- How exactly do you practice it?

Type your text here.....

- Do you think it is important?

- Yes

- No

References

- Aguiar, A., 2009. Tutorial on agile documentation with wikis. In: Proceedings of the 5th International Symposium on Wikis and Open Collaboration. ACM, p. 41. Cited on p. 2.
- Ambler, S., 2002a. Agile modeling: effective practices for extreme programming and the unified process. Wiley. com. Cited on p. 15.
- Ambler, S. W., 2002b. Lessons in agility from internet-based development. *Software*, IEEE 19 (2), 66–73. Cited on p. 33.
- Ambler, S. W., 2008. Agile software development at scale. In: *Balancing Agility and Formalism in Software Engineering*. Springer, pp. 1–12. Cited on p. 25.
- Baptista, J., 2008. Agile documentation with uscrum. In: Proceedings of the 26th annual ACM international conference on Design of communication. ACM, pp. 275–276. Cited on p. 27.
- Biju, S. M., 2010. Agile software development methods and its advantages. In: *Technological Developments in Networking, Education and Automation*. Springer, pp. 603–607. Cited on pp. 25 and 62.
- Boehm, B., 2002. Get ready for agile methods, with care. *Computer* 35 (1), 64–69. Cited on p. 33.
- Briand, L. C., 2003. Software documentation: how much is enough? In: *Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on*. IEEE, pp. 13–15. Cited on p. 29.
- Buckl, S., Matthes, F., Neubert, C., Schweda, C. M., 2011. A lightweight approach to enterprise architecture modeling and documentation. In: *Information Systems Evolution*. Springer, pp. 136–149. Cited on p. 34.
- Chen, J. Q., Phan, D., Wang, B., Vogel, D. R., 2007. Light-weight development method: a case study. In: *Service Systems and Service Management, 2007 International Conference on*. IEEE, pp. 1–6. Cited on pp. 26 and 31.
- Clear, T., Jun. 2003. Documentation and agile methods: striking a balance. *SIGCSE Bull.* 35 (2), 12–13.
URL <http://doi.acm.org/10.1145/782941.782949> Cited on p. 34.
- Cockburn, A., 2006. *Agile software development: the cooperative game*. Pearson Education. Cited on p. 15.

- Dagenais, B., Ossher, H., 2006. Guidance through active concerns. In: Proceedings of the 2006 OOPSLA workshop on eclipse technology eXchange. ACM, pp. 60–64. Cited on p. 34.
- de Souza, S. C. B., Anquetil, N., de Oliveira, K. M., 2005. A study of the documentation essential to software maintenance. In: Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information. ACM, pp. 68–75. Cited on pp. 34 and 66.
- Deemer, P., Benefield, G., Larman, C., Vodde, B., 2010. The scrum primer. Scrum Primer is an in-depth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective, available at: <http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf> 1285931497. Cited on p. 28.
- Dooms, K., Kylmäkoski, R., 2005. Comprehensive documentation made agile – experiments with rapid7 in philips. In: Bomarius, F., Komi-Sirviö, S. (Eds.), Product Focused Software Process Improvement. Vol. 3547 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 224–233.
URL http://dx.doi.org/10.1007/11497455_19 Cited on p. 34.
- Forward, A., Lethbridge, T. C., 2002. The relevance of software documentation, tools and technologies: a survey. In: Proceedings of the 2002 ACM symposium on Document engineering. ACM, pp. 26–33. Cited on p. 77.
- Fowler, M., Highsmith, J., 2001. The agile manifesto. *Software Development* 9 (8), 28–35. Cited on p. 26.
- Golafshani, N., Dec. 2003. Understanding Reliability and Validity in Qualitative Research. *The Qualitative Report* 8 (4), 597–607. Cited on p. 13.
- Goodwin, C. J., 2009. *Research in psychology: Methods and design*. John Wiley & Sons. Cited on p. 8.
- Hayhoe, G. F., 2012. Iso standards for software user documentation. In: Professional Communication Conference (IPCC), 2012 IEEE International. IEEE, pp. 1–3. Cited on p. 78.
- Highsmith, J. A., 2002. *Agile software development ecosystems*. Vol. 13. Addison-Wesley Professional. Cited on p. 15.
- Hoda, R., Noble, J., Marshall, S., 2010. How much is just enough?: some documentation patterns on agile projects. In: Proceedings of the 15th European Conference on Pattern Languages of Programs. ACM, p. 13. Cited on p. 2.
- Hunt, J., 2006. Agile methods and the agile manifesto. *Agile Software Construction*, 9–30. Cited on p. 29.
- Kajko-Mattsson, M., Jan. 2005. A survey of documentation practice within corrective maintenance. *Empirical Softw. Engg.* 10 (1), 31–55.
URL <http://dx.doi.org/10.1023/B:LIDA.0000048322.42751.ca> Cited on pp. 34 and 66.
- Kobayashi, O., Kawabata, M., Sakai, M., Parkinson, E., 2006. Analysis of the interaction between practices for introducing xp effectively. In: Proceedings of the 28th international conference on Software engineering. ACM, pp. 544–550. Cited on p. 29.

- Kothari, C., 2009. *Research methodology: methods and techniques*. New Age International. Cited on p. 7.
- Kuppuswami, S., Vivekanandan, K., Ramaswamy, P., Rodrigues, P., 2003. The effects of individual xp practices on software development effort. *ACM SIGSOFT Software Engineering Notes* 28 (6), 6–6. Cited on p. 29.
- Mazni, O., Sharifah-Lailee, S.-A., Azman, Y., 2010. Agile documents: Toward successful creation of effective documentation. In: Sillitti, A., Martin, A., Wang, X., Whitworth, E. (Eds.), *Agile Processes in Software Engineering and Extreme Programming*. Vol. 48 of *Lecture Notes in Business Information Processing*. Springer Berlin Heidelberg, pp. 196–201. URL http://dx.doi.org/10.1007/978-3-642-13054-0_18 Cited on p. 34.
- Neergaard, M. A., Olesen, F., Andersen, R. S., Sondergaard, J., 2009. Qualitative description—the poor cousin of health research? *BMC Medical Research Methodology* 9 (1), 52. Cited on p. 7.
- Patton, M. Q., 1990. *Qualitative Evaluation and Research Methods*. Sage, Newbury Park. Cited on p. 12.
- Prause, C. R., Durdik, Z., 2012. Architectural design and documentation: Waste in agile development? In: *Software and System Process (ICSSP), 2012 International Conference on*. IEEE, pp. 130–134. Cited on p. 25.
- Rüping, A., 2005. *Agile documentation: a pattern guide to producing lightweight documents for software projects*. Wiley. com. Cited on pp. xi, 2, and 16.
- Sauer, T., 2003. Using design rationales for agile documentation. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*. IEEE, pp. 326–331. Cited on p. 15.
- Saunders, M. N., Saunders, M., Lewis, P., Thornhill, A., 2011. *Research Methods For Business Students, 5/e*. Pearson Education India. Cited on pp. 8 and 9.
- Schwaber, K., 2004. *Agile project management with Scrum*. O’Reilly Media, Inc. Cited on p. 28.
- Shore, J., et al., 2008. *The art of agile development*. O’Reilly Media. Cited on p. 16.
- Sommerville, I., 2001. *Software documentation*. *Software Engineering* 2. Cited on pp. xi, 17, 18, and 20.
- Stettina, C., Heijstek, W., Faegri, T., 2012. Documentation work in agile teams: The role of documentation formalism in achieving a sustainable practice. In: *Agile Conference (AGILE), 2012*. pp. 31–40. Cited on p. 35.
- Stettina, C. J., Heijstek, W., 2011. Necessary and neglected?: an empirical study of internal documentation in agile software development teams. In: *Proceedings of the 29th ACM international conference on Design of communication. SIGDOC ’11*. ACM, New York, NY, USA, pp. 159–166. URL <http://doi.acm.org/10.1145/2038476.2038509> Cited on pp. xi, xiii, 35, 36, and 38.

- Taulavuori, A., Niemelä, E., Kallio, P., 2004. Component documentation - a key issue in software product lines. *Information and software technology* 46 (8), 535–546. Cited on p. 18.
- Wong, K., Tilley, S., 2002. Connecting technical communicators with technical developers. In: *Proceedings of the 20th annual international conference on Computer documentation*. ACM, pp. 258–262. Cited on p. 30.