NTNU

Norwegian University of
Science and Technology

A Case Study of Usability Design in Ocean for Petrel Development at Schlumberger Information Solutions.

Linn Veronica Flataker

Master of Science in Communication Technology
Submission date: June 2009
Supervisor: Lill Kristiansen, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Schlumberger is in a special position when it comes to developing one of their biggest software, called Petrel. In addition to develop the software themselves, they allow other people to develop plug ins that looks like they are a part of the real software. They do this by creating an API that the plug in developers use to create the plug ins into the real program. This thesis is about how Schlumberger Information Solutions (SIS) creates and maintains an API framework that services other developers to produce software plug-ins that in turn service their end-users.

In this thesis I will examine some of the challenges of the developers of the API regarding usability.
Some of the challenges for the developers of the API are:

- How does the Ocean developer understand what the user of the plug in needs?
- How do the developer develop for good usability, both for the user of the API, and also the user of Petrel
- How to help the users of the API to satisfy usability requirements for their end – users
- How the new adaptation of a Scrum work process are affecting the team's ability to deliver better usability

Assignment given: 26. January 2009
Supervisor: Lill Kristiansen, ITEM

Preface:

"The good life is inspired by love and guided by knowledge"

- Bertrand Russell

The quote above is how I see life. A person has to gain knowledge to make meaning of her life, but without love inspiring it, life has no sense.

These four years I have spent studying on NTNU have been educational. I have evolved as a student, and as a person, and my thirst for knowledge has increased. During these years, my fascination with people and use of computers started. That was when I realized that even if you made software with the best functionality in the world, it would be useless if it didn't have good usability.

I was searching for a topic for my master thesis on Usability in the fall of 2008, when I met Susanne Schumann from Schlumberger. I had a telephone interview with her, and Danny Dykes, my project manager to be. With Danny's help, we shaped the assignment together, taking a starting point in their situation at Schlumberger Information Solutions.

I would like to thank Schlumberger Information Solutions and the team in Oslo and Stavanger for helping me and guiding me in my writing of my master thesis. I thank especially Susanne Schumann, and my team manager Danny Dykes for making this possible. They were incredibly helpful, and aided me whenever I needed something. The Ocean team was always helpful and asked any questions I needed answering. In addition I want to thank my professor, Lill Kristiansen, she was there to help me start in the right direction, and came with useful feedback. I would also thank my family and friends who were there for me all through my education and the writing of my thesis, backing me up, and helped me get as far as I have.

Abstract

Schlumberger is a service oilfield provider company which also develops their own software for their clients. They are in the special situation where they have developers who develop an API, which is then used to make plug - ins into their software, Petrel. So the challenges I have encountered and researched are about usability, and how the developers for the API, develop for usability, not only the API usability, but also the usability of the plug - in users. The developers have work that are characterized by complexity and that is dynamic, so that they have to evolve rapidly to keep up with the demand.

This research is going to study the case within Schlumberger Information Solutions and come up with some ideas to ease the process to help find solutions to help develop for better usability.

Some of my findings are that Scrum actually can help with the usability if implemented correctly. Because of Scrum, the communication within the Ocean team has increased. Another way to make the API more usable is to add more example code so the API becomes self – explanatory. Increased user input from the end user is another way to add to the usability for the Petrel application, and testing with users earlier on in the process not only when the product is finished.

After I present my findings, I discuss the results, the validity of them, and also the reasons to why Schlumberger doesn't do it like that today.

Contents

Explanation of words and abbreviations used in my thesis:

SIS –Schlumberger Information Solutions

Ocean – A framework used to make plug - ins to software called Petrel

Petrel – A seismic to simulation software produced by Schlumberger

API – Application programming interface

Framework - a basic conceptual structure used to solve or address complex issues.

SMR – Software Modification Request

FTR – Formal Technical Review

UI – User Interface

SCRUM – A type of work process

Product Owner – the one in the team that are responsible for the requirements

List of figures in the paper

Chapter 1 - Introduction

Usability may be defined in many ways, because it is a part of human perception. Here is three examples on definitions of usability:

1. One of the developers I interviewed gave me this example for definition of good usability: The coca cola bottle. It is argued that it has good usability, because, when you look at it, you just want to grab it and drink it. It has something to do with the shape of the bottle. This is a well known example used by teachers to illustrate usability. Just by looking at it, you know instinctively what to do with it. All things around us have some kind of usability.

2. "Usability is a **quality attribute** that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process." (Nielsen, 2009)

3. API Usability: Usability Engineering describes all of the parts of designing an interface in a way that users will be able to learn, comprehend, and remember easily. API Usability is the intersection of good usability design and good programming techniques. (Koelle, 2007)

Sometimes an application can have good usability for one type of clients, while it has not good usability for others. Still, there are some principles that are important to follow that ensure good usability for most of the users. This means that usability is highly individual.

"Designing an object to be simple and clear takes at least twice as long as the usual way. It requires concentration at the outset on how a clear and simple system would work, followed by the steps required to make it come out like that – steps which are often harder and more complex than the ordinary ones. It also requires relentless pursuit of that simplicity even when obstacles appear which would seem to stand in the way of that simplicity." (Schneiderman and Plaisant, 2005, Chapter 1)

This quote is still valid, more than 30 years later. Nowadays the word usability is a buzzword, every company claims that they strive for good usability, but why is it that so many software projects still fail? Since the quote above still holds, where it says that it is hard to develop for good usability, the developers often have to weigh the cost of using some extra time for the usability, or not. The cost is money, it takes more time, and the employees use more of their time doing that than start on a new task or requirement. The advantages of good usability will not be seen on the development stage, it will come later. First of all, if it has good usability, it is more likely not to be a total failure and not be used at all. It is not good if all that hours and developers are wasted just because the usability is not good enough. Then the cost would be high for a product that isn't going to be used at all.

Chapter 1.1 starts with the assignment text as it was given to me by my project manager, Danny Dykes. It was originally formed by questions, but I took the liberty in changing it into bullet points to make it easier to read. In chapter 1.2 I discuss the goals that I set before starting the research. Chapter 1.3 describes what kind of interpretation and constraints I had to put on the assignment. My incentives and why this topic is interesting is described in chapter 1.4. In chapter 1.5 I present an overview of the rest of the thesis and what kind of topics I will present in it.

1.1 The assignment

Schlumberger is in a special position when it comes to developing one of their biggest software, called Petrel. In addition to develop the software themselves, they allow other people to develop plug - ins that looks like they are a part of the real software. This thesis is about how Schlumberger Information Solutions (SIS) creates and maintains an API framework that services other developers to produce software plug-ins that in turn service their end-users, and how to maintain usability in this regard.

In this thesis I will examine some of the challenges of the developers of the API regarding usability.

Some of the challenges for the developers of the API are:

- How does the Ocean developer understand what the user of the plug - in needs?
- How does the developer develop for good usability, both for the user of the API, and also the user of Petrel
- How the new adaptation of a Scrum work process are affecting the team's ability to deliver better usability

1.2 Goals for the thesis

With background in my assignment from my project manager, I developed some goals to aim for while researching.

1 Get an understanding for the developers situation regarding usability

2 Try and find solutions to help Schlumberger Information Solutions improve their process to develop for better usability

The solutions I have is more suggestions that can help the team improve the communication with the end user, and make the whole process easier to understand what the end user wants. In addition I point to other efforts that can be done to ensure that usability is good. I will use the information I gather about the situation together with literature that I have studied, and discuss my findings at the end of the thesis.

1.3 Limitations of the assignment

As a researcher I had to make some boundaries for my thesis to be able to gather enough material and to also be able to analyze it. Time was the main constraint for me here, the thesis are supposed to be done in 20 weeks the last semester of my Master's degree. In addition there is money to consider. Schlumberger has been very willing to let me interview their developers and take time from their schedule so that have not been as much an issue. It could be, but since I got to decide how many weeks I could be with them in their offices, I could have been there more if I needed that.

In addition, the field researched here is very extensive and complicated, so I had to focus on the questions; how does the developer understand the end user of Petrel's needs, how to develop for better usability, and how the work process Scrum affects usability. I concentrate on how the developer understands their users, and if Scrum does make it easier or worse to develop with good usability.

The things I didn't look into is how the part of the team situated in Abingdon worked, or how the support team could give feedback on usability, because they know and see where the users have the most problems. This is interesting topics that I wish that I had time to research.

With that said, I have to think about many things that can affect the usability in this case mainly because the setting is so complex that it is difficult not to try and take in as many causes as possible to get an overview of the situation.

1.4 Why this is an interesting topic

Usability has become a, what we call today, buzzword. Every developer has heard of the word and has some kind of perception of the word. The topic of usability in the traditional way, where we have one or more developers and clients is interesting enough, but when I talked to Schlumberger and heard about the problems and challenges they had regarding API usability and how to develop for good usability for the end user, I was very interested. They not only have to think about the API user usability, but also the usability of the plug - in that the API user make. There is unfortunately not much theory written on exactly this subject, and this case is rather special. But it can probably be applied to other companies that develop frameworks. This is because even though Schlumberger is in a special position where they have both the API users and the end users in the same company and can talk to them, other companies that develops API's must also think about both the end user usability as well as the API usability.

How do the developers in Schlumberger consider developing a suitable GUI when they develop for other developers? How does the communication happen between these? It is difficult for the developers here in Oslo to know what the end users need if they don't get the complete requirements specifications from the end users.

API usability is also an interesting field for me, I have not been studying that directly earlier in my study, but most of the principles of regular usability can be applied. Not regarding where to put boxes or what color things should be, because this is an API that doesn't have a user interface directly, but all applications have usability in some way. The usability we talk of regarding API is typically naming of methods, documentation and good example code so that it is easy to know how to use them and what to do with them. If the naming is incorrect, or ambiguous it will be difficult for the user to use the API to figure out what to do, and the efficiency of the production will decrease.

One of the reasons I find this research area important is just because there is little research on it. I actually get to look into a problem area that not many people have been able to look at. In addition I think that probably some of the findings I get are applicable to other areas where you have API developers, because they should also think about usability for their users, and the end users.

1.5 The structure of the thesis

This chapter was an introduction to my thesis and described the assignment and some of the problems for a developer in Schlumberger regarding usability. I also described the limitations of this thesis, because when researching a topic, there are always more to do, regarding time, interviewing people etc.

In the next chapter, 2, I get into the research methods, and talk about the two main research methods used. I will describe my choice of method for gathering my data and the reasons behind my choice. I also discuss the validity of my research and the methods that have been used for data collection.

In chapter 3 I describe some theory regarding usability, both usability of user interfaces, but also usability regarding API usability. Most of the usability principles can also be applied to API usability in cases of how to ensure it and how to approach the problems.

Since this is a case scenario, I use chapter 4 to give the reader insight in the area of research, and describe how things are done in Schlumberger. This includes how the requirements travel from the user to the developer, and also how the different teams in Schlumberger Information Solutions cooperate.

In chapter 5 I write about Scrum and explains what it is with theory before I talk about how it affects the team in Ocean and the changes they have made because of it. The reason I chose to have this part in another chapter than the theory chapter is that in addition to explain what Scrum is, I also describes the changes of Scrum in the team and how Scrum has affected their work process. In chapter 6 I discuss the findings regarding Scrum, so that I have all my findings in chapter 6

All my findings are in the chapter 6. Here I present the most important issues that I found was interesting to study and to look at.

The discussion and analysis I do in chapter 7 where I discuss the validity of all my findings, and also some implications that needs to be considered in this case. I also present some improvements that can help Schlumberger Information Solutions in making the process and talking to their clients easier.

In chapter 8.1 I write my conclusion of the thesis, and in 8.2 I discuss further research I would do in this field if I didn't have the limitations and restrictions I had for the thesis, and what parts of research I would look further into.

Chapter 2 – Method

Choice of research method depends on the type of research that has to be done, and what kind of data that is needed to be gathered. In addition, it depends on the time the researcher has to disposal, the researcher's preferences and experience, and access to material. The choice of method needs also to consider advantages and disadvantages of the method chosen and how these can influence the finished report and results. All research methods have disadvantages, but the goal is to choose one that fits best into the type of research you want to conduct.

In the first part of this chapter I am going to describe the two main categories of research methods and differences between them, in chapter 2.1, and then I will describe the research method I chose and the four most usual ways to acquire data within that research method; interviews, observation, document analysis in chapter 2.2. After that, in chapter 2.3, I will get into the reasons for my choice of research method. In chapter 2.4 I describe how the data was collected, mainly by interviews and observations, but also by document analysis and training.

Chapter 2.5, 2.6, and 2.7 describes my choices of informants and the credibility of the research. In the end of this chapter, chapter 2.8, I reflect upon my role in this research and in what ways I could influence the result of the research.

2.1 Quantitative vs. qualitative methods

One usually divides the main research methods into quantitative and qualitative research.

"Quantitative data means data, or evidence based on numbers" (Oates, 2006) Quantitative research concerns mainly data that is numerical, and is good for acquiring statistics. This is used if you need to look at an extensive field, and you can gather data that can be statistical analyzed. You usually gather quantitative data by experiments and surveys, but it is also

possible to use other data collection methods as interviews and observation, and then analyze it quantitative. This kind of research is often easy to analyze because the result can be calculated and other researchers that question you result can try it out themselves. (Oates, 2006)

Qualitative research is usually gathered by case studies, action research and ethnography (Oates, 2006). The goal in using qualitative method is to be able to describe and research a topic that is not possible to reduce to just numbers. It explains the question "how" rather than "what" or "where", and it makes it possible to get people's opinions, experiences and behavior and analyze that. Qualitative research can also contain numbers and numerical data, but looks also on other explanations than just the mathematics can explain.

Qualitative research gives a more nuanced picture than quantitative research, and that is sometimes more useful, depending on the topic that are researched. It is useful when you need to study a phenomenon in its natural context. Examples of ways to gather data using qualitative methods are interviews, but mostly not structured ones, observations and qualitative document analysis (Dalland, 2000). It is supposed to be possible for the researcher to try and not be a part of the situation that is researched, but this is an ideal, because it is impossible for the research not to be colored by the researcher experiences and opinions, even if it is unintentional. Advantages of qualitative research is that it gives the researcher an option besides study fields that can be reduced to numbers, and the result can be a lot of things, text, drawings. There does not have to be one explanation, qualitative research have possibilities to accept more than one explanation to a phenomenon. One disadvantage of qualitative studies is that it is possible to gather a vast amount of data, and it is difficult to know where to start the analysis. In addition, as mentioned above, it is impossible for the result to be objective, and not a result of the researcher's opinions. (Oates, 2006)

The form of the data is often in a format that makes it difficult to analyze, such as audio tapes from interviews or figures etc. (Oates, 2006) I overcame this problem by transcribing all of my interviews from audio tapes to written data so that it would be easier to analyze.

In addition, I used pen and paper to analyze, it is much easier when you see the common factors before you and can cut and paste instead of on the computer

Quantitative and qualitative research it two very different research methods, and the reason for choosing one over the other depends mainly on what the topic for your research is and what kind of data that needs to be gathered. Since I am researching a topic that is not easily reduced to and analyzed using numbers, because it is such a complicated field, my choice became naturally qualitative research.

2.2 Qualitative research

Qualitative research is a field of inquiry that crosscuts disciplines and subject matters. Qualitative researchers aim to gather an in-depth understanding of human behavior and the reasons that govern such behavior. The discipline investigates the why and how of decision making, not just what, where, when. Hence, smaller but focused samples are more often needed rather than large random samples. (Robson, 2002)

It is possible to divide qualitative research into three types of different methods. You have case studies, ethnography and also grounded theory:

- Case studies: Is mostly used when you need detailed knowledge about one or more certain settings. Methods used to gather the data is observations, interviews and analysis of documents that the researcher has gotten access to.
- Ethnography: Is used when the researcher needs to get an in – depth understanding of a field, e.g. how a company works, how people live, behave and see the world from their standing point. Observation and interviews are usually used to collect data.
- Grounded theory: This is when you first start to collect the data, and then form your theory after analyzing them. Methods to follow in this theory are to first gather the data, then analyze it, then form your theory and test it. If the test proves the theory wrong, you start over again from the beginning. (Robson, 2002)

Even if the research is to study a case scenario, this does not make it immediately clear that the research method should be case studies. I could use either ethnography or grounded theory to research my scenario. However, I choose case studies as the research method because I believe that is the most appropriate to my case scenario.

2.2.1 Case study

When you conduct a case study, the case that is being researched, can be a person, a company or part of it or almost anything. During the years, there have been endless definitions of a case study, but I am going to use Robsons's (2002) definition of it:

"A case study is a well – established research strategy where the focus is on a case in its own right, and taking its context into account. It typically involves multiple methods of data collection. It can include quantitative data, though qualitative data are almost invariably collected."

A case study is as stated above suited to study a phenomenon in its right context, and this fits with my case; I am going to study how the developers of the Ocean API thinks about the usability for their end users when they develop their API. Also, as mentioned above, the usual way to collect data when doing a case study is interviews, observations and document analysis.

2.2.1.1 Interviews

Interviews are one way to gather information. An interview is basically that the researcher gets contact with one or more people that know anything about the case or the topic that are being researched, and the researcher ask questions about it that the interviewee

answers. In another way, you could say that an interview is a special form of a conversation between two or more persons. The interview is always planned beforehand, and has a special topic to discuss, so it cannot be compared to a regular conversation between two people. (Oates, 2006)

You have different types of interviews; it is normally divided into structured interviews, semi – structured interviews and unstructured interviews.

- Structured interview: This kind of interviews has pre specified questions in a pre – set order. The questions are the same for all the interviewees, and the extreme version of it is a questionnaire where the interviewer does the writing.
- Semi –structured interview: This has also predefined questions, but the order of the questions can be exchanged during the interview. You can change the wording of the questions, and you can add some questions, while omit others during the interview.
- Unstructured interview: This type of interview can be informal, and the researcher typical has a wide area of interest, and lets the interviewee talk about what comes into his mind. But he has to steer the conversation back onto the subject if the interviewee gets off the topic of discussion. (Oates, 2006)

The two last types of interviews, semi – structured and unstructured is mainly used in qualitative research. These are more suited for in – depth knowledge about a field than structured, and are used in case studies. There is also a fourth type of interview, which is called informal. This is typically conversations during lunch and can be used to clarifications and such, but this cannot be used as a sole method of research. These kinds of interviews give the informants more possibilities to talk about their own experiences and attitudes about the issue discussed.

There is one type of information that is difficult to get with using questionnaire or phone interviews. That is the knowledge the interviewee conveys with his or hers body, and one

of the reasons I chose interviews over other data collection methods was to also try and understand that type of knowledge to get a whole view of the situation and the interviewee.

This presents a challenge for the interviewer, to try and get an understanding both of the explicit knowledge, but also of the tacit knowledge. Interviews are a well known and used method to collect the necessary data. It is flexible and easy to tailor to the researcher's needs and interests. In addition to observation, an interview gives us valuable knowledge, such as the tacit knowledge described above. This is not the case with questionnaires or document analysis, these ways of collecting does not convey information like tacit knowledge. When you meet the interviewee in person, you can change your questions as you go along, and make sure that all the answers are answered properly, and in case of misunderstandings, you can correct them and give an explanation of it. And non verbal communication can help you understand the meaning of what the interviewee says, or reflect upon whether or not he tells the truth. This gives the research or thesis a rich and detailed type of data.

When using qualitative research method, as stated above, it is common to use open questions, which mean basically questions that cannot be answered in one word, or yes or no. This gives the interviewer an advantage, because if he or she finds anything interesting he can probe deeper and ask more follow up questions to find out what the interview object meant by the statement etc.

One of the disadvantages of this kind of interview is that it is easier for the interviewer to lose control over the direction the interview is going and that it probably will be harder to analyze when there is less structure. In addition it is hard to prove the reliability, what was actually said, and how much is due to the interpretation of the researcher. Does the researcher have a bias towards any results? If he finds something that prove his or hers theory wrong, is he likely to omit that information, either intentionally or unintentionally?

In addition, the interviews take a lot of time. The researcher needs to prepare for the interview, conduct it, and then preferably transcribe the contents so that it becomes easier to analyze. It doesn't only take of your time; it takes of the time of the interviewee also. The value of an interview depend a lot of the skill and expertise the researcher has in that area. If the researcher is not experienced, he or she can ask not skilled enough questions so the point of the interview is wasted, or don't pay enough attention so that he misses important points made by the interviewee. This can be corrected by the interviewer trying to have some rehearsed interviews with family and friends to gather some experience before actually conducting the interviews regarding the thesis. It is also a drawback that the interviewee tries to answer as he think we want them to, and omit information that can put them in a bad light. But that kind of information is hard to get anyway, and is not special for the interview method of acquiring data at all.

Interviews are one way to gather data for analysis, but it is also possible to use it in combination with other means, for example document analysis or observations, or both.

2.2.1.2 Observation

Observation is when a researcher observes a certain group or person in a specific setting. Researcher often uses observation as a research method, because then they can see what people actually do, instead of just hearing them tell you what they do. People often has a misunderstanding of how much they one specific action, so there is almost always a difference between what people say they do, and what they actually do. Observation can also be used so that the researcher gets a better understanding of how things work in the specific case. It is difficult to ask questions about the atmosphere or the setting and other things, it must be experienced.

The recipe for observation is as following; you meet the people you are observing, you write field notes during observation, and after that, you analyze the field notes so that you can interpret what you have experienced.

You have some types of observation to choose from. First of all, it ranges between structured and participating observation. (Oates, 2006). Structured observation is often prearranged, and the researcher decides in advance which events he wants to observe. In the other end of the scale, you have participant observation, where the observer is part of the setting he is observing. It can range between these two kinds, so we talk about the grade of each instead of two different types of observation.

When observing, either the researcher is a part of the group being observed or not, he should reflect on what kind of influence he is on the group being observed. Either way, it is most likely that the group will modify their behavior a bit, either deliberately or subconscious. When it is structured observation, the researcher is not part of the group, and will not influence the result as that, but the group will be aware that he is present, and therefore try and be on their best behavior. When we talk about the participant observation, the observer is actually a part of the group and will definitely influence the result himself. This can be an advantage if the researcher try and understand as much as possible about the group being observed, but can also create a danger that he gets too into the group and forgets to see things objectively and with new eyes.

In addition to the grading between observation methods above, there is also a distinction between covert observation and overt observation. (Oates, 2006) Covert observation is when the group that is being observed doesn't know that they are being observed at all, and the researcher goes "undercover". This is a smart way to go if you really want to see how the setting is, but it can be dangerous if the people being observed finds out, and also it is difficult to write field notes without raising suspicion. Overt observation is the one I have talked about this far, but the disadvantage of this is that the researcher has to be aware that his presence will affect the result.

Advantages of observation are that you get to see the setting for yourself and more impressions is gathered, than in interviews for example. It is easier to observe the setting yourself than ask the questions about everything, and not all things is possible to describe either. You can see people's attitudes and responses to events that happen, and register them. It is quite cheap to execute, you only need permission and pen and paper. The data

gathered from the observation can help you understand better other information that you have gotten e.g. from documents or interviews. It gives the researcher a good picture of the real life, and can be more correct than interviews because people often have another perception of the situation than it really is. (Oates, 2006)

One huge disadvantage of observation is that the researcher unintentionally affects the results. It is therefore important that the researcher try and put themselves out of the equation and ask themselves "How would people act if I wasn't there?" As long as the researcher is aware over the potential problem and can reflect upon that according to his findings, it is the best way to do it. As mentioned above, there is also a danger to "go native" where you become one of the people you are trying to observe, and lose perspective. Also, there is a danger to misinterpret the material you analyze, as it is for every other type of data collection within qualitative research except for document analysis. (Robson, 2002)

2.2.1.3 Document analysis

Document analysis is when the researcher analyses a document that has a connection to the research that he is doing. This can be a manual, a book, letter, notes from a meeting etc. This is the only data collection method that doesn't actually affect the material being analyzed. Both in interviews and observations, the researcher is in the situation and can affect the results. However, the interpretation of the document analysis is of course subjective by the researcher. The usual way to analyze a document is to analyze the contents qualitatively.

When researching a document, there is two important issues to think about; the validity of the document, and to analyze it according to the context it was written in. Document analysis is often used in addition to other data collection methods as interviews or observation to gather a complete overview of the situation or case.

Advantages with document analysis are that the material and results is not affected by the researcher in any way, besides the analysis of it. In addition, this is a permanent medium,

and the researcher doesn't have to cope with variable things like the mood of their interviewee for example. This means that it is possible for another researcher to do the same research to check whether the results are valid. You don't need any other person than yourself and access to the material, so bother other people and using their time is avoided.

Disadvantages could be that the analysis is still pretty subjective according to the researcher, and not all the material is written to be read by outsiders of the company so it can be difficult to understand and interpret the material.

When researching a case it is possible to view each data collection method as its own viewpoint, or glasses. To get a best possible result of the whole picture, it is useful to use more than one pair of glasses or data collection method.

2.3 Reason for choice of research method

When you choose a research method, it is important to be aware of the strengths and the weaknesses of the chosen method, as this can influence your result. As long as the researcher is aware of the weaknesses, he can try to do everything in his power to diminish their effect.

My work in this case is mainly to talk to developers and understand how they work and interact with their clients, and analyze this according to existing theories. In addition I will try and come up with some solutions for how Schlumberger Information Solutions can improve their process and help the developers think more about usability when they develop.

The data I need to gather to do my thesis and my work for Schlumberger is qualitative. Therefore, qualitative research is the only reasonable choice for me. In addition, this choice is natural since it is a special case that I am going to look into. Yes, you can maybe draw out some general points that can be used in other companies that are doing API development, but mostly the things here are special for this case. The methods I have used

for my data collection have mainly been semi – structural interviews, but I have also used observation, informal interviews and a kind of document analysis, which I describe in the next part of the chapter.

I not only need to know what the developers are doing, but also why they do it this or that specific way. I need to gather a broad specter of information, and not specific information of a small subject, and this can only be achieved using qualitative research. In addition I have experience in conducting interviews and doing observations, because I had a course teaching this, and also both observed and interviewed people for my project assignment during fall 2008. It is safer to choose a research method that is well known by the researcher and that he knows its strength and weaknesses and how that might influence the results.

One of the he reasons I chose qualitative research is its main strength and that is the possibility to gather a lot of detailed data in a diverse environment. There is also more freedom in choosing the path of the research, if I find something that are interesting, it is possible to spend more time pursuing that than in quantitative research, where you decide beforehand what to research.

2.4 How the data was gathered

I was at Schlumberger Information Solutions office in Oslo, Røa, a number of times, and was there one or two week period of time. In total, I was in Oslo for 5 weeks during the spring 2009 between January and April, and in Stavanger visiting the team there in three days in March.

I started out in Oslo getting to know the developers creating the Ocean Framework, and I also started to learn the end – product Petrel software to get a feel for the user experience. Petrel is an extremely large computer program, so I didn't have a chance to learn it very

thoroughly, it takes years to be really skilled in using this software, that is how big it is. The software users are geophysicists and geologists, so I didn't have the competence in those areas to use the program. In addition I learned to use the Ocean Framework, a little by myself, and I also attended a one week course in Ocean to learn it. In this course we learned to use the framework to create the plug – in ourselves, and it helped me a lot when I started to try and understand the situation of the developers and users. I got an understanding on what problems the users of the API encountered and how difficult it was to get the plug - in to look like the real program.

2.4.1 Observations

Before I started to observe the team I had to decide which type of observation I should conduct. The main types of observation are covert observation and overt observation. I chose overt observation, the safest method, which means that the team knew I was observing them, but this method has also some complications.

The observation was done in an unstructured manner; I just tried to write down as many details as possible to learn about the environment of the developers. This was so that I had everything in writing, in case something would later seem important that I didn't think was important then and then I had it for later analysis. I attended a two – week sprint meeting which usually lasts for an hour every other Monday where I listened in on the questions and concerns the team discussed. In addition, I got to overview four daily stand – up meetings, and one FTR – formal technical review where the team and their customers goes through a requirement and discuss difficulties and problems regarding that requirement.

I gathered most of the data and information while I was there at Schlumberger's office. This is not a formal way of acquiring data, but I got a lot of information by asking questions in an informal setting, e.g. over lunch. This is called informal interviewing. It was very informative to actually be there for one or two weeks, because the developers saw

me as a part of their group in the end and I didn't feel like an intruder, and I didn't think that they saw me as that. In addition, the possibility to be there for a week at a time made me gather a lot of impressions I wouldn't have been able to get if I was just there one day for a meeting or something else. I got a feeling for how the developers worked and the atmosphere there at the office. A lot of the sources I have for information about the setting in Schlumberger are things I have heard and seen and put together myself, I don't have an "official" written source for all the things I know about the team and their work process.

2.4.2 Interviews

In addition to observation, I also got to do interviews with the developers, and other persons I thought was relevant and I did 12 interviews in total. Eight of them were in Oslo, and four in Stavanger. I mostly interviewed the developers, but also product owners and other people that could help me in my search for answers. This includes also the usability engineer for the end product Petrel.

My interviews was semi structured, I had a list of questions to ask, but I also followed up with new questions on the spot if it was something I didn't understand or wanted the interviewee to explain more. In appendix A there is a list of questions I asked and in addition consent for them to sign, that explained their rights and the persons that were going to have access to the uncensored material. The reason I asked them to sign it was to make them feel free to answer truthfully to my questions, and don't be afraid that someone from their workplace got a hold of the information they gave me. I didn't ask every interviewee the same questions; they changed regarding the role in SIS. Some of them had expertise in some of the areas I researched, but not all of them, so I just asked the questions relevant to the interviewee.

I was in Stavanger for three days with the Ocean for Petrel team there. Here the team consists of four people with computer science backgrounds and geophysics background. They work on Ocean on the geophysics side of it, in Oslo they work on UI and

Visualization and the data domain. I used the first day in Stavanger to get to know the team and observe them work and on the daily morning meeting and prepare questions for the interviews.

In addition I interviewed them on Thursday and Friday. In this team, I got to choose the people to interview all by myself. Since they were just four people, I chose two of them. The other two that I didn't interview I had discussion with them during lunch break, so I knew a little about their viewpoints, so I didn't feel it was necessary to interview them also. They hadn't prepared anything for me, so I chose two of the team members there to interview, and in addition, I got to interview the Petrel Usability Designer and the old product owner who had a lot of experience in the product owner role, since the current product owner for Ocean is new in his position.

I chose to record all of my interviews beside one that I didn't get permission to record, and as soon as possible after the interview I started transcribing the material.

From the beginning, it was scheduled for me to go to Abingdon in England to interview and observe and interview the team there, but as I saw a lot of similarities in the team in Stavanger and the team in Oslo, I decided I had gathered enough data. If I had more time for my thesis, interviewing the team in Abingdon would be one of the things I would do. The reason I chose not to go and see them was a combination of practicalities, the team there was in a middle of moving to another office, and that I felt I had the data I needed. In addition the teams in Oslo, Stavanger and Abingdon have the same challenges and does mostly the same work, but within different domains. The reason that I that the process with the requirements is the same in Abingdon as in Oslo and Stavanger is that it is the same product owner that has contact with the different teams and that assign the different tasks. So up until the team, the requirements come from the same source, but it would have been interesting to see if the implementation of Scrum was different from the Norwegian team.

I chose to do all of the interviews in English, even the ones with Norwegian interviewees.

This is because of simplicity in regard to my thesis, I write it in English and it would be difficult to translate from Norwegian to English without losing some of the meaning. I also recorded all of my interviews besides one on tape to make the work later easier and for me to concentrate on other things than writing.

2.4.3 Document analysis - training

I have not been doing document analysis in the sense that I described it earlier in this chapter. None of my final results descend directly from document analysis, but it has been some of training at Schlumberger where I have been going through documents to get familiar with the case setting. I chose to describe document analysis above, because it is a part of the main methods of collecting data.

I didn't analyze that much documents from Schlumberger, but in the beginning there was a lot of stuff to read up on because this is a field that I was not that acquainted with from school. I started out with trying to get to know the software, Petrel, with reading a manual about it and trying it out, playing with it to get a feel for how the user interface and what kind of tasks it is used for. After I got used to Petrel, I didn't learn thoroughly how to use it, it is very big software, I started out trying to learn how to use Ocean, the framework to create plug – ins myself. This was also task that was big, and Ocean as a framework has very many opportunities. I found out how big both of it was and how complicated it is to work with it. I followed a tutorial myself in the beginning, but I struggled a bit, so they signed me up for a one week training course in Ocean that I attended. This was very lucky, because the course had just been moved from some other country in Europe to Oslo, so I was lucky that I could attend, and that it also was the same week that I was there.

2.5 Selection of informants

My project manager chose the people for me to interview for my first interview round in Oslo. This is called selection via a gate keeper. I got to interview almost all of the

developers in Oslo, and if I wanted to interview more people, I got the permission to go around and make appointments with people myself. I got a good representation of the developers in Oslo, since I got to interview almost everyone, which were five out of six Ocean developers. The consultants in Budapest are in some ways part of the team in Oslo, but I didn't interview any of them, so in total in Oslo and Budapest, I interviewed 5 out of 8 developers. In Stavanger, I interviewed 2 out of 4 developers, which is 50%.

The developers and employees in Schlumberger differ in age, nationalities, specialties, education and gender. This is not a homogenous collection of people, so I tried to interview as many different developers as possible. In total, I interviewed 12 persons, and out of these it was 7 developers. I interviewed 3 women and 9 men and of them only 5 Norwegians, one from China, one from India, three from the US, one from France and one from China. And the age ranged from about 30 and up to 50, so it was a diverse group in age also. When it comes to experience, it also ranged from a couple of months to 20 years for Schlumberger. I also interviewed developers with different expertise in the Framework, e.g. some were usability and visualization expert, some were experts on the data part of the API, and other again in the seismic domain.

Besides developers, I interviewed the product owners, both the former and the current, because the current just started in his job to get a more nuanced picture of the process from the end user to the developers. I also interviewed one from the commercialization team, the Petrel usability architect and the Ocean for Petrel architect.

Some of the interviewees were mostly for me to get an insight in how things were done in Schlumberger Information Solutions and to get to know the field. This is a very large area with many developers and users, so I used a lot of time getting to know how things were done in this complicated and large company.

The selection process was not based on statistics, I let the theory decide, I made the selection strategic, because I needed people to tell me about the topic I needed information

about. In addition was availability another issue, the ones in Oslo I interviewed when I was there at the office, and I made an extra trip to Stavanger to interview the developers there.

Regarding the selection of interviewees, I got a good sample of the developers, and the others I interviewed more for background research to understand the field discussed. I didn't get to interview the people from the team in Abingdon because of time limit both in regarding to fly over to England to conduct the interviews and regarding the time it would take to analyze the material. In addition, there were some logistical problems; the team over there was in the middle of switching to new offices, so it would be inconvenient to some level. It could have been interesting to see how things were done over there, at least regarding Scrum. I also needed the time to perform analysis on all the data I had. If the case was that in Norway, just Norwegians worked, I would have taken the time extra to interview the people in England to check for differences, but the process is mostly the same there as in Norway. In addition, it is more foreigners than Norwegians working here and in Stavanger, so they have many different nationalities, and as described above, I interviewed people from many different countries, so diversity was already reached.

Before the interviews started, I explained the reason for the interview and the topic of my thesis. All the people I interviewed were told in advance that if they felt uncomfortable and wanted to quit the interview that was possible. I ensured them also that the material they provided me with would be made anonymous.

Also I created a form where I and they signed, that stated who had access to the material before it was made anonymous.

Written consent to be interviewed for the thesis:

" The Art and Science of Usability Design in API Development: A Case Study of Usability Design in Ocean for Petrel Development at Schlumberger Information Solutions."

The undersigned are hereby informed that

- Participation in this interview is voluntarily and that I can quit at any time
- All the information I present will be anonymized
- The conversation will be taped and deleted when the project is finished
- Only the interviewer, the professor Lill Kristiansen and the censor will have access to the material before it is anonymized

I hereby agree to participate in this survey.

Name: _____

Email: _____

Signature informant: _____

Signature interviewer: _____

Linn Veronica Flataker
Email: linnvero@stud.ntnu.no
Tlf: 930 55 173

Lill Kristiansen
Email: lillk@item.ntnu.no
Tlf: 977 27 227

Figure 1: Written consent form, created by the author

2.6 Analysis of the material

The reason for using data collection methods as interviews and observation etc is to gather data. But a research doesn't stop with just collecting a vast amount of data. When you analyze the collected material, it goes under a change from observations to science. But before such a transition is possible, the material needs to be in a format that is is possible to analyze. The reason I tried to transcribe it right after the interview was that I then had many other impressions than just the words said fresh in memory. I had the feeling from the interview, did the flow of the interview go well, and things like body language that is impossible to record on audio tape. I interviewed 12 people, so I had a little less than 11 hours of tapes in total to transcribe. The transcription along with my notes from the interview and the notes from my observations made the foundation of my analysis.

Since I had many of the interviews close in time, after each other over two days, at least my initial interviews, I had to wait with transcribing some of the interviews. During transcribing, I wrote also down my own notes about the subject, laughing, pausing, did the interviewee seem nervous, that kind of things. This was some sort of an initial analysis of the interviews and material. This helped me analyze and interpret the meaning of the words he said, was he laughing nervously, were we on a subject that were uncomfortable for the interviewee?

After transcribing, I started the analysis itself. First, I printed out all the material of transcription, and started reading through them with no particular goal in mind. While I did that, I noted down cues for myself, and after that I sorted through them while gathering answers on the same topics for themselves. I actually tried using free software on the computer for this, color coding the text and cutting and pasting, but found out that I rather liked working with paper and cutting and pasting. Some of my topics where I divided the text were "Scrum", "Usability, "API usability", etc. When I created out the subjects, I

along with the analysis looked at the theory I read prior to the analysis, and my assignment to answer the issues that were raised in my assignment text.

After a while I started recognizing patterns in the answers and gathered them together. I have to keep in mind that it is easier to notice tendencies that comply with my own predefined perceptions, so there can be some connections that I haven't seen. (Thagaard, 2003) When I later present my findings, I will make it clear which are my interpretations, and which are the words of one of my interviewees. I will also quote my interviewees to make it clear who's meaning it is, them or mine.

I promised all my developers full confidentiality so what I have done in this case when I quote them, I use "developer 1", "developer 2" said "Quote." In addition, I didn't use "she" in any cases, despite the fact that there were also female developers; I chose to use "he" since there was not that many females, and thus the quote could be led back to the one that said it. I tried also to omit quotes that were position specific, so that the topic would not point to a specific person or a developer.

2.7 Credibility of the research

"Troverdighet er knyttet til at forskningen utføres på en tillitsvekkende måte." (Thagaard 2006)

This quote says that "Credibility depends on that the research has been done in a way that can be trusted"

The researcher has an obligation to ensure the reader that the results and findings can be credible. A big part of this comes from the collection of data and how they were treated afterwards. It is important that the material that makes up the result of the research is material direct from the informants, e.g. quotes etc and not just the researcher's interpretations of the data. (Thagaard 2006)

One of the ways for me to ensure credibility of the research that I have conducted is that I have taped and transcribed all the interviews. I cannot release that in its full form, but I can make it anonymous and use quotes. Also, the analysis have been done carefully, as to not try and add meaning or interpretations to the data, but it is possible that I subconsciously focused on some parts of the research while omit other parts or findings that I didn't see as important.

2.8 The role of the researcher

When using a research method as qualitative, it is important to be aware of its own influence on the setting that is being researched.

I also have to be aware that the kind of person I am and the choices I have made during this time writing on my thesis can affect the result. I am a person with a history, certain kind of upbringing and values, and these I have with me all the time, and color the way I see the world. The kind of person I appear to be has very much to say for whether the persons I am talking to trust me etc. These kinds of things may seem unimportant, but are actually essential to make the interviewee feel comfortable. It can be things that are so small that they don't even notice it consciously, but it helps with the overall impression. Everything I say and how I act can influence the way they see me at Schlumberger and that again will influence how they treat me. Am I a person that invites to trust, or not? Can something in my behavior tell them things about me that can make them careful not to tell me the truth?

As a researcher I have to reflect up on the fact that I could influence the team in Oslo with my presence as discussed above in the chapter when discussing qualitative research. When I observed the team during their daily stand – up meetings, I have to be aware of my position. They knew that they were being observed, and could either consciously or unconsciously adapt their behavior to something they thought I needed to see, or try and make them self look as good as possible. This could mean that the team changed their

behavior accordingly so that I didn't get the right impression of them. They could feel that it was uncomfortable that they were being observed, and they maybe tried to act as they thought I wanted them to act. Or if they thought that my job was to report back to their boss, since I was introduced by him, they could just put on a "play". Even if the team didn't actively think about me when I was there, they probably would change a little bit of their behavior, either it was intentionally or subconsciously.

In a way, I got some way around the disadvantages of overt observation with the arrangement I had in Oslo. I had my desk in an open area because it was no available offices. In the area where I sat, the team had their meetings, daily stand up meetings and their two week sprint meetings. So I didn't hover over them, or stood there watching, I simply sat by my desk and heard what they said and saw what they did with very little impact on them. I think that they also sometimes forgot that I was sitting there, because I was always sitting there, and they got used to me being there.

In addition, I observed many of the daily standup meetings, and they did almost the same every day, the team discussed what they did since last etc. It is a regular meeting, and if one or more of the team members started acting very differently than normal the other would probably react to that. But of course, a developer could choose not to say something that he or she otherwise would have discussed with the others because of me.

Also the fact that it was the project manager that introduced me, I didn't know any of them before this, could color the answers I got from the interviews and also their behavior during observation. The fact that I was introduced by the project manager could lead the team to believe that I was sent by him to "spy" on them. This could lead to answers in the interviews that they thought I would want to hear, or that their boss would be satisfied with. To try and convince them that that was not the case, I gave them a paper to sign where it said that the information they gave me could not be traced to them and that the data would be made anonymous.

I also recorded the interviews, this can be a two – edged sword, some people can be very careful with what they say "on record", but then I got the chance to focus on what they said

instead of writing everything down. This made the interviewee feel more comfortable again, because he or she felt that I was listening to them and what they said instead of just be occupied with writing. And in addition, I could use my full attention at them and ask more follow up questions on important subjects or areas when I felt that I could find something more or get a better understanding.

I actually had one interview where I couldn't tape the interview because he was an extern and didn't want to be on the record, so I had to write everything down. I think this affected both me and my interviewee a lot, when I had to write everything all the time, and I didn't have the chance to transcribe and analyze the data later. I, of course wrote everything I could think of down right after the interview, but I didn't remember whole phrases, so I felt that I didn't get as much out of it that I could. Some of the problem could be due to all my other interviews were taped, so I had little experience in interviewing and writing at the same time. Before I started this thesis, I just have done one interview for practice during a course at NTNU and that was with a person I knew very well, so this was a lot different, and I was a bit nervous at my first interviews. This can also have affected my interviewees in a negative way if they noticed it, but I tried not to show it. If I was nervous, could that mean that they should be also?

I tried also to tell them that I was not out to see how well they did their work tasks, just notice how and what they did. I also told them that I didn't have enough in – depth knowledge of the field to judge how well they did their work. Many times when the team delved into details and discussed that, I didn't have the background that they had in the field, so I didn't know exactly what they said. But I noticed what kind of problems that arose and what kind of things they used the stand up meeting to discuss.

I gave them forms to sign so as to promise them that it was in any case, only me and my professor that would have access to the material when it was not censored. This was one of the things I did to make sure that I got as sincere answer in the interviews as possible, and that they didn't just answer like they thought their boss or I wanted them to answer. As long as I am aware that my findings can be colored by me or my presence it makes me treat my findings with a bit of caution.

Interview guide for the interviews I conducted during my master thesis.

Before the interviews, I prepared thoroughly, and tried to make the interviewees as comfortable as possible. I arranged the seating so that the angle should be most preferable as I studied, the interviewer and the interviewee should sit 90 degrees for each other. In addition I dressed appropriately so that my appearance should not affect the setting and tried to dress as them, casual. If I had appeared to be another person, e.g. dressed all in sweats that would have affected the interview. I wanted the interviewee to feel as comfortable as possible. In addition, we used their meeting room for the interviews and this was good, because it was a familiar place for them, so it was more likely that they open up, and tell things as they are.

These are the questions I mainly asked my interviewees, the developers. When it came to the others, the commercialization person, product champions and Petrel Usability Designer, I used these as a starting point, but asked them more about their field of expertise.

1. Name
2. Age
3. Education
4. How long have you worked for Schlumberger?
5. What is your role in the team?
6. What do you think about when I say the word ”usability”
7. Do you ever think about usability when you develop?
8. How does a developer understand what the end-user of another developer really wants? Do they need to?
9. What are the most common usability issues that you come across during your developing?
10. Do you have any ideas to make it easier to develop for better usability?

11. Do you feel that going over to Scrum has helped the communication between team members? In regards to usability, do you think that it will be easier now to develop for better usability?

12. How does a developer design for usability, not just in API consistency, but also to aid a developer to satisfy usability requirements for their end-users?

Chapter 3 – Theory

The field in question here with API development is quite complex. There is not as much theory on the subject of usability regarding the both the API user and the end user, and that complicates some of my research. I chose to take a starting point in basic theory on things I consider in my thesis, and then later on describe theory from other more complex areas.

In this chapter I am going to explain some basic concepts, in background of literature that already exists. In the first chapter I am going to define usability, and then in the next chapter I will define usability in API context in chapter 3.2 and give some guidelines in developing for good usability. In chapter 3.3 I discuss the importance of usability testing, and in chapter 3.4 I discuss tacit knowledge in regards to the choice of interview methods in chapter 2.

3.1 Usability

Usability has almost as many definitions that there are people. I asked the developers of the Ocean API what they thought of when I said the word usability, and everyone said something different. Usability is also individual from person to person. What may seem as good and logic usability for one person doesn't necessarily mean that it would be the best solution for another person.

The developers in SIS thought about usability as API usability or usability for their end product, Petrel. Most of them mentioned "ease of use" or consistency as criteria for good usability. In this chapter I talk about usability in general, before I describe API usability in the next chapter.

Usability can be looked at as a combination of criteria that must be fulfilled to accomplish good usability. Usability consultant Jakob Nielsen and computer science professor Ben Shneiderman use these concepts or criteria when they describe good usability:

- Time to learn how to use a system: How easy is it for users to accomplish basic tasks the first time they encounter the design? This is important for this case, because the longer it takes a developer to learn new features; it costs money if it takes longer than it has to be.
- Speed of task execution: Once users have learned the design, how quickly can they perform tasks? The more effective the application is to use, the more the company earn in form of efficiency which again leads to more money saved.
- Memo ability: When users return to the design after a period of not using it, how easily can they re establish proficiency?
- Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- Satisfaction: How pleasant is it to use the design? Does it give the user a good feeling about it, or does it fill the user with frustration that it doesn't work? It is easier for a user to get back to an application if is satisfactory to use it.
- Consistency: This is an important factor, because the users need it, so that they won't have to learn how to do the same thing over again if they have been gone. E.g. a website that a user visits each month. If it didn't have consistency, the user has to learn new ways to do the same thing every month.

The six criteria above are taken from (Schneiderman, Plaisant, 2005).

3.2 API Usability

There is a difference between "regular" usability of a GUI, Graphical User Interface and API usability. Most of the principles of how to get good usability apply also to API usability. But there is not a GUI included, so that can be a little hard to understand how the usability is here, but I will claim that every application or software has usability. As long as it is going to be used by humans and it is either; simple and clear to use, or it is not. Either way it has some form of usability, good or bad.

“Usability in an Ocean context I think more of making it easy for the plug-in developers to understand the API and to use the API to do what they need to do without having to read a lot of documentation and see a lot of example code. “ Developer 1.

This means taking advantage of the five usability issues I mentioned in the chapter above. In this chapter I will discuss ways to make an API more usable.

3.2.1 How to make the API usable?

According to chapter 3.1, to make a good API usable, you have to make it easy to learn by users, make it possible for them to execute tasks efficiently, the error rate should be as small as possible and that subjective user satisfaction is good enough. (Schneiderman, Plaisant, 2005)

There are three things that can ensure usability and effectiveness in an application; the number of tasks, task frequencies and the task execution time. (Jaques, 2004) describes four ways to do this;

- Have good visibility in the API; this means that the best solution to accomplish this is to have the possible states, actions and the alternatives for possible actions exposed. It is no need to remember how to do things, when it is already there.
- Have a good conceptual model: If the developer provides abstractions about the system that are complete, this helps the user get a correct mental image of the system.
- Have a natural mapping between actions and their results.
- Good feedback of the results of the user’s actions.

When it comes to the first criteria in the list above, visibility, there are a few guidelines to follow to achieve this. Good visibility helps on the learning time, and the retention time, which is the time it takes for a person to forget how to do a task using the API. There are many types of information about an API, and the most important information should be clearly visible. Visibility has also some drawbacks, one in particular; if you have an

extensive API, should everything be visible? The more visibility of everything, the more mess you can make. So the thing here is to just make the most important features visible, and let the others be.

Good conceptual model is the user's image of the system. If this is complete, it will give the user a good overview of the API and how to perform actions. The natural mapping between actions and results counts also for other usability, not just API usability. It should always be clear to the user what happens if they perform a certain action. Natural mapping is the easiest and best way to do this, because then the user has something to map the actions with. These criteria above helps as guidelines when developing an API with good usability.

3.2.2 Other issues to think about when it comes to creating a usable API

API Usability is in the intersection of good usability design and good programming techniques. Below, I discuss some other things that are also critical to create a usable API. When you're developing a software library, you are actually designing an interface. In fact, it's an Application Programming Interface, or API, which your users - other developers - will need to learn, comprehend, and remember easily. (Dave Koelle, 2007)

- The developer has to start out by thinking of the end product. What is the purpose of the API, what do you want to accomplish? It is a good idea to create a sample application before you start developing. This will help you refine the methods that you intend to define and give the developer a better understanding of the application. Test-Driven Development is also important both when it comes to regular usability and API usability.

- Create a compact API. The developer should aim for few code lines which does what it is supposed to do. It should be clear, accurate and easy to get an overview of. The fewer lines of code the user has to write, the better!
- Be absolutely correct. The users are relying on your API to do what it says it will. This creates a sense of expectation from the user, and if this is not correct, it will create frustration for the user and errors.
- One point says also that a developer should make himself available for comments from the user. If the developer takes advantage of the feedback for himself, he can use it to make the API better.
- Be aware of errors from the beginning and correct them. This is important, because the earlier you get to the errors, the less work is lost.
- When something happens, e.g. an error occurs, it is important to explain what went wrong. "Wrong input for method XX", instead of something like "oops, something bad happened."
- When developing and evolving and API it is good to be clear on what things that have changed since the last version. It is difficult to fix something when it is first released because many make code relying on your API. This is why it is so important with user feedback to minimize the occurrence of such errors.
- The presentation of the API is part of the success. Why should anyone use it, if they don't understand it? It should have a clear description of it, either in a manual or on a website. (Koelle, 2007)

3.2.3 User input

One of the ways to ensure good usability is to have user input before and during implementation and that the clients give feedback so that the developers can improve the API to the next time if a mistake is done. In this case the user makes the requirements to the API developer, but it isn't always that good communication. The ideal is that the end user and the developer meets and discusses the requirement if there are any problems. This is done because of the cost, it costs a lot of money to take people out of their job of production and have them test new things.

When you have an API, and people that use it, use it effectively, you save time and money for the company that have the software. In addition, when the API is easy to use, you also save time on that for the hundreds of developers using it.

3.3 Testing usability

One of the ways to achieve good usability is to test it thoroughly both during development to guide the development and after. This is a way to try and find usability issues, to improve them before the development is finished. (Myers, 2004). During the years usability has been thoroughly tested through a vast amount of application during the years, and the engineers have principles as those above to use as guidelines, but usability is a subjective issue and should be tailored to the users of each application. The usability of an API is of course different to the usability of software with graphical user interface, but also the users are highly different from the two different applications. The user of the API is skilled developers, and the users of Petrel are mostly not educated in IT.

Testing should be done by inviting the clients who are the real user of the application and have a test session with them where you give them tasks to accomplish and note whether the test persons have any trouble finishing the task etc. The issues raised here, should then be implemented in the development so that the developers can improve their product. (Schneiderman, Plaisant, 2005)

3.4 Tacit knowledge

The knowledge that you get out of an interview, like semi – structured or structured can be divided into two groups. You have the words and meaning of them which are called explicit knowledge, and then you have the other type, tacit knowledge, which is what we don’t say, body language, intonation etc. (Argyris and Schøn, 1996).

When doing a case study, it is important to get to know the attitude of the interviewee and what they do, think and feel in addition to the facts that they say explicitly. The words that they say is important enough, but sometimes, the tacit knowledge is just as valuable. These are things that you cannot ask a person, because it is difficult to put into words etc. We can describe the ratio between tacit and explicit knowledge as an iceberg, what they express with words (explicit knowledge) is about 10% of total communication which is the part of the iceberg which is above water, while the tacit knowledge is about 90% and is much more difficult to convey and interpret and is the part of the iceberg below sea level.

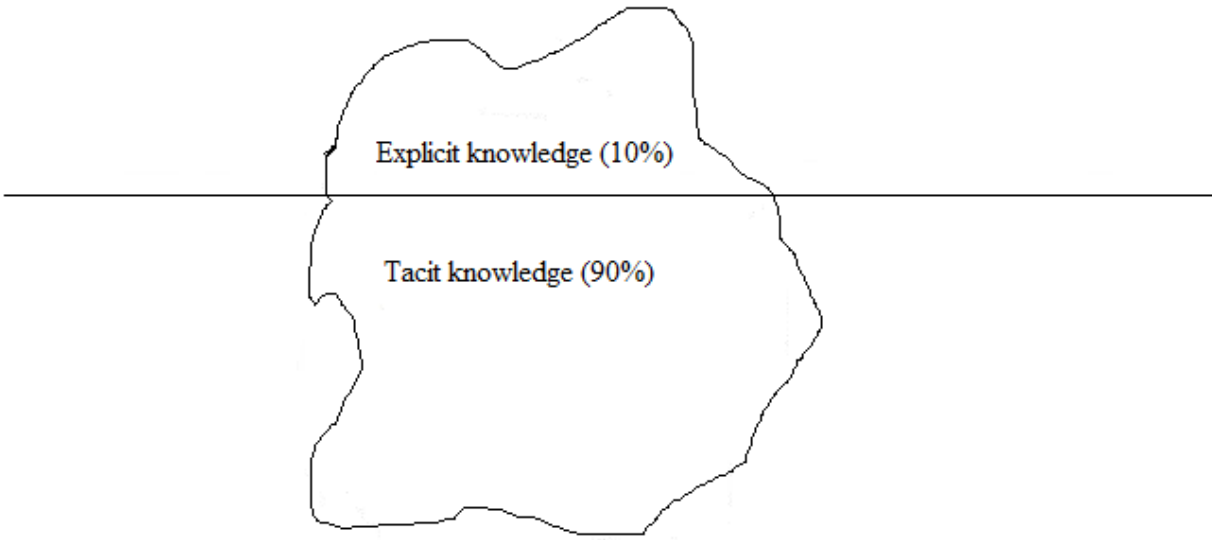

Figure 2: Tip of the ice berg metaphor

The theory about tacit knowledge is added here because it helps me explain in chapter 2 why I chose the data collection methods that I described in chapter 2.

Chapter 4 – the case: Schlumberger Information Solutions

“Schlumberger is the leading oilfield services provider, trusted to deliver superior results and improved E&P performance for oil and gas companies around the world. Through our well site operations and in our research and engineering facilities, we are working to develop products, services and solutions that optimize customer performance in a safe and environmentally sound manner.” (Schlumberger,1)

One of the divisions of Schlumberger is SIS, Schlumberger Information Solutions. Schlumberger Information Solutions (SIS) is an operating unit of Schlumberger that provides software, information management, IT infrastructure, and services. SIS enables oil and gas companies to solve today's tough reservoir challenges with innovative workflows enabled by open collaboration and comprehensive global services, step-changing the effectiveness of E&P teams. Through our technologies and services, oil and gas companies can improve business performance, reduce exploration and development risk, and realize the potential of the digital oil field. (Schlumberger, 2)

To be the leading oilfield service provider, they need technology to accomplish their goal. They sell big software called Petrel which is a Seismic to Simulation Software. (Schlumberger, 3) They have clients using Petrel within Schlumberger and they have also clients outside of Schlumberger. To cope with all the requirements that is posed by so many clients, they decided to use a framework so that the clients that own Petrel could make their own plug - ins and make it look like the plug - in is a part of Petrel. This framework is called Ocean for Petrel. The framework provide additional functionality to the standard Petrel workflow and are provided as an executable that adds the module to your existing Petrel install. (Schlumberger, 4)

Ocean is an open software development environment that offers seamless integration of your intellectual property into the Petrel mainstream workflow. It allows your developers to focus on innovation, rather than infrastructure. The environment leverages the modern

.NET tools, and offers stable, user-friendly interfaces for efficient development. (Schlumberger, 5)

Schlumberger is a company that has done some controversial decisions in the past. Schlumberger chose to develop an API which their clients could use to develop Petrel further. In addition, their clients get full ownership of their plug - in and can sell them in competition with plug - ins that Schlumberger develops itself. This is a bold decision, but can be useful in the long run, with a lot of content customers that gets software that they can use to their needs, and tailor it after that need.

Some of the references in this chapter has been taken from Schlumberger web sites as there was no other references for literature on this subject. However, the pictures that are referenced to this website are not always visible, because they change dynamically. If the figure is not on the page that it is referenced to, the reader has to update the page some times to make the exactly figure appear.

In this chapter I explain why I chose Schlumberger Information Solutions as a research case, and then I explain the setting at Schlumberger. In chapter 4.3 I discuss usability issues regarding the software Petrel.

4.1 Why Schlumberger

I have heard about Schlumberger early in my education and I was interested in the interdisciplinary problem description between IT, humans and geology/geophysics etc. I got in contact with them during Woman and Technology day, where they invited female technology students from all over Scandinavia and Great Britain to Oslo. There I got to watch a demo of this software called Petrel, and they had this interesting topic for me regarding usability in the API versus the end user. I also wanted to write for a company and to get a feel for how the problems with usability are dealt with in real life, not just theoretically in books as I am used to from my education.

So I started out in Oslo with the Ocean team there, the team is spread in Oslo, Stavanger and Abingdon, and in addition, they have contracted out some work to some consultants in Budapest and Bangalore.

In addition this was an assignment that was split in two parts. I got a chance to see what API usability is about, which I didn't know what was until I started researching the field before I started the thesis, and it was also about Scrum. Scrum is a buzzword nowadays, and I think that it is good for me to know the advantages of Scrum regarding usability, since that is what I want to work with later in life, and I most likely will work in projects that use Scrum.

4.2 The setting

In Schlumberger, they develop a type of software called Petrel which is a seismic to simulation software. It delivers fast, intuitive and productive geophysical interpretation, geologic modeling, reservoir engineering, and innovative domain science for more accurate reservoir characterization. (Schlumberger, 3)

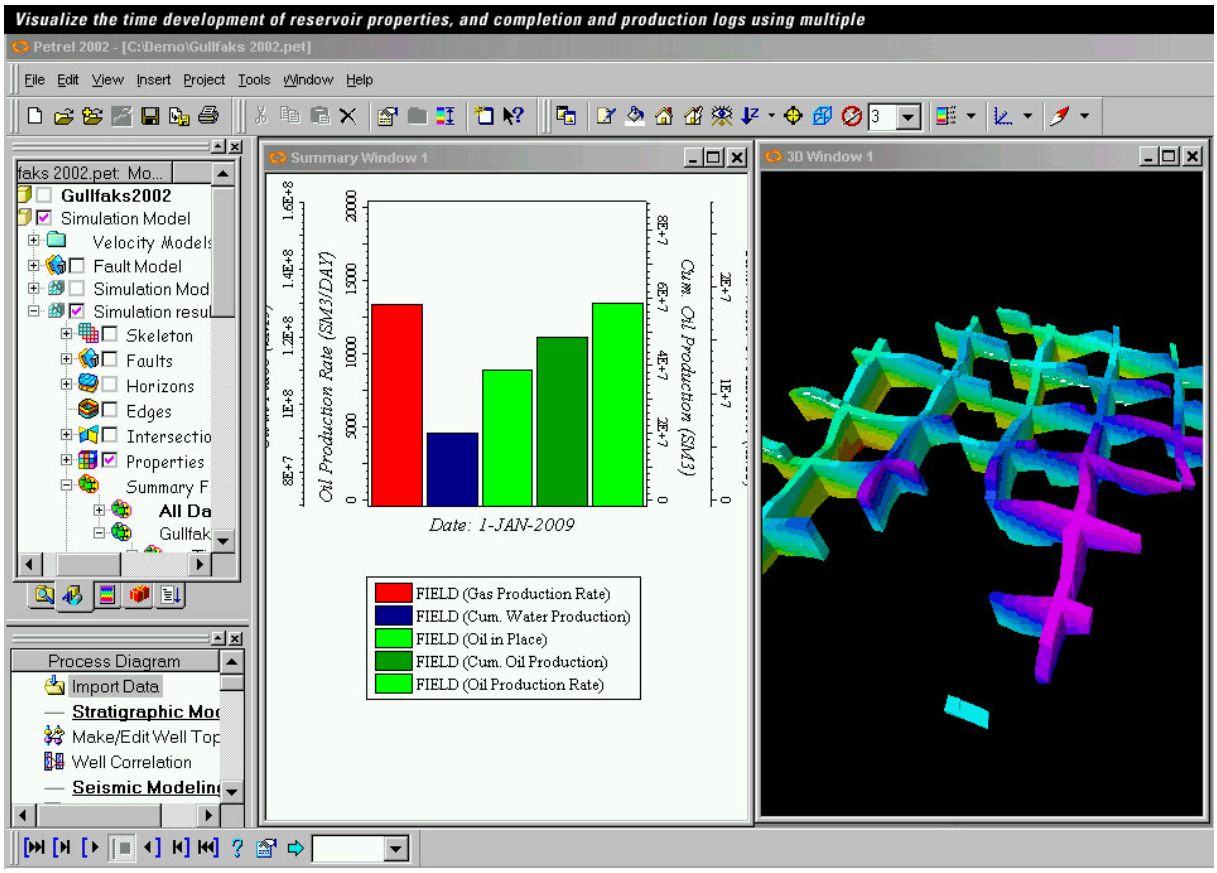

Figure 3: Screenshot of Petrel:
<http://www.slb.com/content/services/software/geo/petrel/simulation.asp?>

The screenshot is included here so the reader can see what type of software Petrel is and that it has endless opportunities, and that this is so big that it is difficult to learn every aspect of it.

In Schlumberger Information Solutions, there are many types and groups of developers, not just for Ocean, but other software also, but in the case I study, you have three different groups of people; the developers of the Ocean framework, the users of the Ocean framework which is also the developer of the plug - in application and the end user which is the user of the end product Petrel. The end users of Petrel are mostly not IT educated personnel; they might be reservoir engineers in geology, or geophysicists etc.

They have developers in both in Oslo and other countries that develop this software Petrel and the new functionality in it, but they have so many diverse types of clients that need different functionality in Petrel. Petrel is used by both customer within Schlumberger, and customers outside Schlumberger by companies like Shell, BGC, Chevron and WesternGeco and many more. Some companies need special features that others don't and to cope with that, Schlumberger uses a framework called Ocean that gives the clients the opportunity to develop new functionality to Petrel themselves.

The clients use Ocean to create new plug-ins with the functionality that they need. Then they have the possibility to run their plug - in from within Petrel, so that the users of Petrel don't even need to know that the plug - in is not a part of Petrel. This makes Petrel very flexible for the customers and they can add new functionality that they need.

4.2.1 How the different groups interact

In this chapter I will describe how the flow of information goes from the end user of Petrel to the developer in Ocean for Petrel.

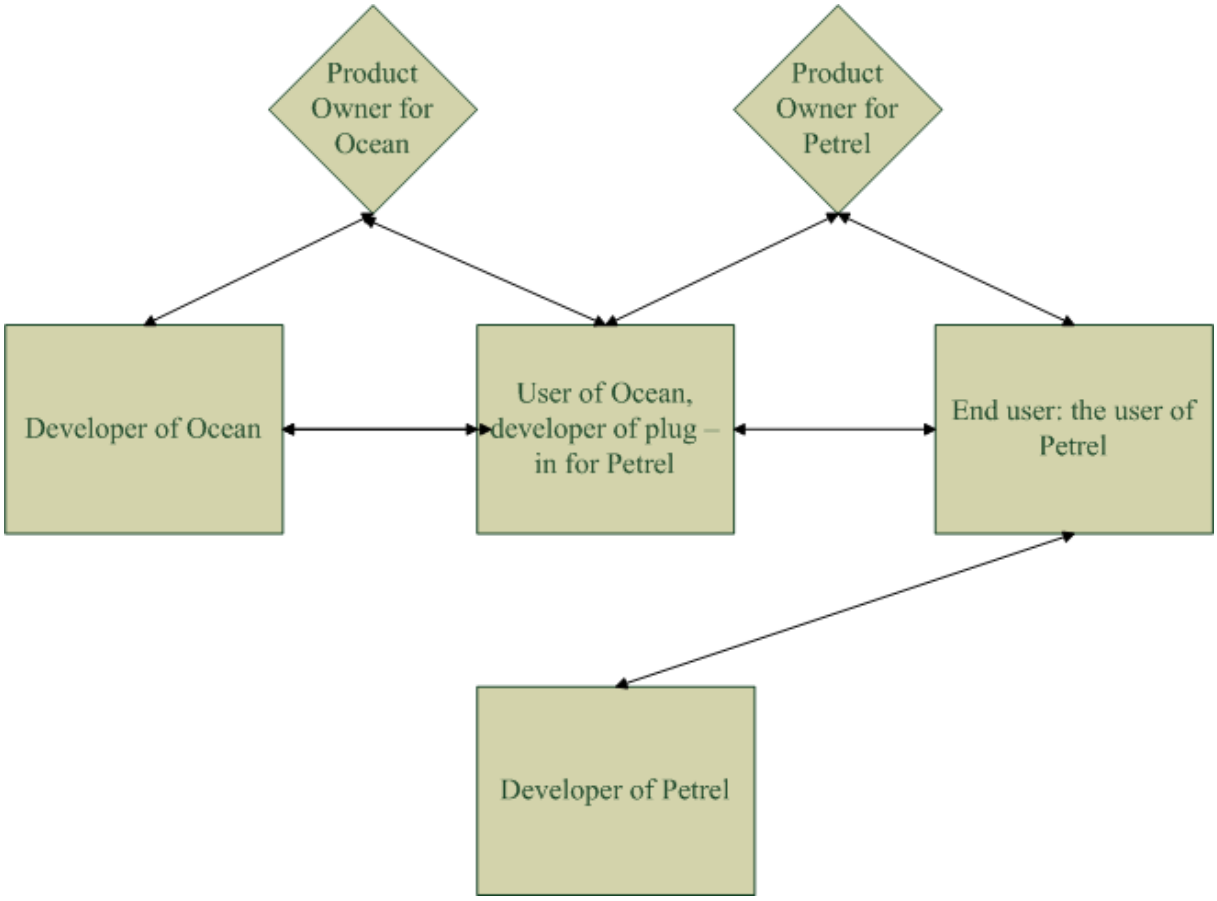

Figure 4: The connection between the developer of Ocean and the end user of Petrel. Created by the author

The figure here shows that the developer of Ocean does not just have to think about their clients and their usability, but also make it possible for the plug – in developers to develop their piece of software with good usability for the end user.

In this thesis I separate between the developers of Petrel and the plug - in developers. The developers of Petrel develop new features that are going to be used in the next version of Petrel. I have not taken these into my account in this thesis at all. I just included them in my figure for completeness. They get their requirements from the users but have a whole other set of criteria to follow. The main area I research is the link between the Ocean

developers, the Ocean users which also are the plug - in developer and the end users of Petrel.

In the thesis I divide between two types of users. You have the API or Ocean user, which is the developer in the API. In addition, you have the end user which is the geophysicists and geologists and the users of Petrel. This is important to keep in mind during reading, that there is two types of end users, but I never use the phrase "end user" about the plug - in developer.

When an end user needs a new requirement for Petrel to do something new, e.g. add a new menu item to a list, he makes his requirement to the plug – in developer. First, they have to make a user story, which is a software system requirement formulated as one or two sentences in the everyday or business language of the user. User stories are used with agile software development methodologies for the specification of requirements (together with acceptance tests). This is used in Ocean because the team uses Scrum as a software process. Each user story is limited, so it fits on a small paper note card—usually a 3×5 inches card—to ensure that it does not grow too large. The user stories should be written by the customers for a software project and are their main instrument to influence the development of the software. (Larman, 2003)

The requirement is posted to the Ocean product owner by the plug - in developer, or communicated via telephone, email of face to face communication. The plug - in developer knows what the end user needs and communicates this to the product owner for Ocean which acts as a middle link. The product owner here is the link between the developers and the plug - in developer. If there is discussion or not enough information for the developer to make the requirement or piece of functionality, he would have to talk to the product owner which then talks to their client again for clarifications.

The clients take contact with the product owner when they need new requirements to be made. In addition, the product owner travels around to the clients to see how they are and if there is something they need. The product owner travels a lot to see their clients in their own environment and to see what new functionality or requirements that they need. It is

easiest to come to an agreement in the beginning of the year so that he sees how much money they have to spend and how many developers he has to use. The team in Oslo was in the spring 2009 in the middle of changing the product owner, before product owner was situated in Stavanger, now the role has changed to one of the persons in Oslo. So the product owner now is new in his job, but during interviews with the developers, they told me that the product owner still acts as a middle link. In some cases the clients contact the developers directly, or the developer contacts the client to figure out something that isn't clear or understandable. This happens mostly in cases where the clients are within Schlumberger, as long as there are clients from outside Schlumberger, they go via the product owner for clarifications.

After the end user and the product owner for Petrel agree on the user story, they go to the Ocean product owner to make a requirement which he conveys via a twiki page to the Ocean development team. In the twiki page, there is a list over the requirements, and their priorities for when they will be developed. One user story is often divided into many requirements for the team to do. These requirements are the one that the team is going to deliver at the end. Then the product owner together with the team agrees on the priority of the requirement; is it very important, or can it wait until next sprint?

The team uses Scrum as a work process so they work in two weeks periods of time called sprints, so they need to set the priority of the requirements. Before they started Scrum they did many requirements in 3months time and it didn't matter which was finished first, because they delivered all the requirements at the same time. So before the team decided when to do what, now it needs to prioritize so that the most important functionality gets done first.

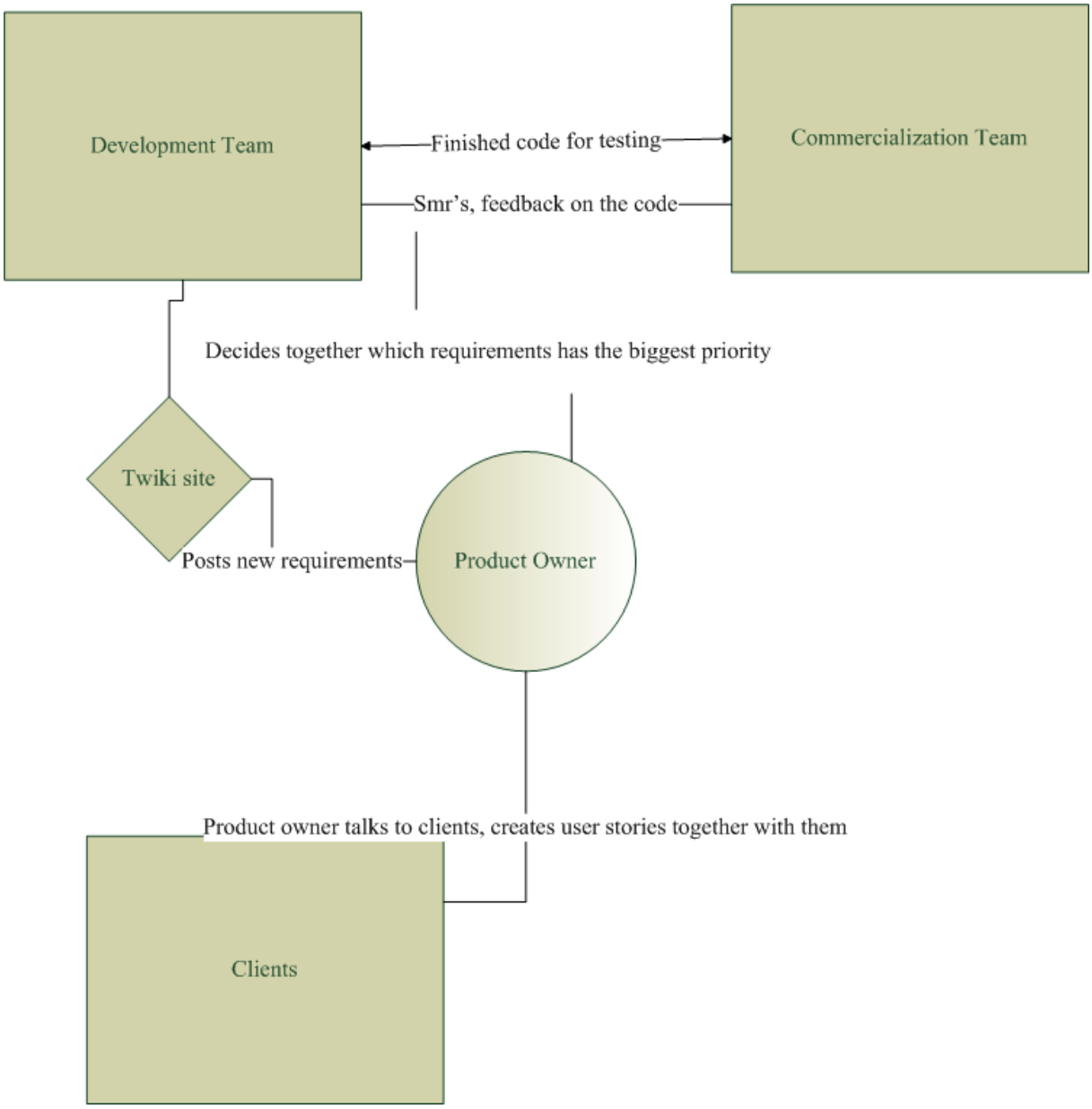

Figure 5: How the different teams work together

You have the teams in Ocean dispersed out in Oslo, Stavanger, Abingdon in England, and the team in Oslo also outsources some of their work to Budapest. The team in Stavanger is quite new and works together with the team in Oslo and Abingdon, but the teams have different areas of expertise. In Oslo they work with visualization and property modeling, in Stavanger they work with seismic, and in Abingdon they work in the Reservoir Engineering domain. In this figure, the three Ocean teams in Oslo, Stavanger and

Abingdon, are called the development team. They all communicate with the product owner, and he decides which team that should work on which requirement. Most of the time, the requirements divides itself naturally to the appropriate team, because the different teams have different specializations. Sometimes one team has very much work to do, while the others don't have that much, and then the product owner working with the engineering manager would try and balance the workload and move the requirement to a team that isn't overloaded with work.

The commercialization team from the figure is the testers, and they get the code from the developers when they are finished with it. They write their own code to test it, and there is usually where the developers start before they are trained to be a regular developer. The commercialization team raises issues and gives feedback on the code so that the developers can fix things. Two of the people on the commercialization team are in Oslo at the office, and the rest of the team is situated in Bangalore where Schlumberger have outsourced that part of the work.

The twiki page is included because that is one of the means that the product owner has communication with the team. He also has meetings, in Oslo he attends the daily standup meetings, and he talks on the phone with the other teams. The thing called SMR in the figure stands for software modification request. This is made if the commercialization team finds a bug or something that needs to be fixed, it can be about the usability of the API, but the commercialization team rarely found bug issues, which was some of my findings. In addition, if the developers themselves find something that needs to be fixed, they can put in a software modification request themselves. This happens occasionally, and if the fix is big enough, the developer sits down with the product owner, and they create a new requirement for it and give it priority.

What the developers need to know about the end user experience they get from the user stories which they get from the plug - in developers and their clients. They need good user stories to be able to understand which part of the API they are going to expose, but they rarely know anything about the user interface of Petrel of the application and what that is going to look like. That is mostly the choices of the user of the API. It should be a high

priority for the Ocean for Petrel developers to try and make it easy for the plug - in developers to make the usability similar to the Petrel usability, but it is difficult, because they never know in advance how it will look like as this is up to the plug - in developer.

One of the main goals of Ocean is to make it self - explanatory. This means that the ideal would be that the customers or users of Ocean could use it without calling support or need outside help of the API. One of the ways to do this is to make a lot of good examples on how to use the Framework API. Many of the developers suggested that this was a way to make the usability and the effectiveness of the API better. As long as the users find out their problems themselves, they don't need to waste their time of the time of the support team and thus get more effective.

4.3 Usability issues in Petrel

The usability of Ocean as a framework is inhibited by a stability promise of two years, which mean that the developers is not allowed to change it during these years. In some ways this will improve the usability, because the users of the API doesn't have to get used to new changes all the time, and in addition have to change their own code accordingly. When something changes, the users of the API have one year to change their code before the method gets changed or deprecated or something else.

The users of Petrel and in general of much software, the people using it are not an IT educated person. Especially when it comes to using Petrel, the users are geophysicists, geologists and petro technical engineers. This creates a big demand for well known good usability. It's not only good usability that a thing is easy to learn and use, but it should have consistency also, so that the users that use it regularly don't have to learn it again when a new version is released. The usability in this case is mostly of consistency. Petrel is used by extremely many people and it's widespread so Schlumberger should weigh up the cost of training someone to use it, and retrain all the others that use it now on a regular basis when they add new features. They also try and have the work flow as similar as

Petrel as possible, so that the user can know what to expect even if they use a plug - in or the real Petrel.

Petrel as a software does not have the same stability promise as Ocean. But the usability engineers of Petrel are very wary of changing things in the software regarding usability. This is because today there are so many people using Petrel, and if its user interface was to suddenly change with new menus and buttons, people would think that the usability decreases. There are of course the new users of Petrel to think about, it should be easy to learn to use it, but the changes should come gradually.

So the usability of Petrel is mostly decided by the old design and old work flow, and the new features needs to look as similar as possible to the old Petrel. Sometimes that is not feasible, such as adding a big component into Petrel that has already been made by another company. It is important for the plug - in developer that the user interface he creates is consistent with the old Petrel. This is a challenge, because you have two types of developers for Petrel, the ones that work for Schlumberger and develops it, and you have also the plug - in developers.

Chapter 5 – SCRUM

In Schlumberger Information Solutions, the teams developing the Ocean API just started using Scrum as a work process. A part of the written assignment text implied that I should take a look at how Scrum affects the development regarding usability, or if it even affects it at all.

In this chapter I have mainly used (Larman, 2004) as a reference. I have not referenced to him all over the chapter, but in some places mainly. In the first chapter here, I describe what Scrum is, and how the work process actually is in terms of examples. My intention here is not to give a full overview to Scrum, it is an introduction, but rather focus on the things that I believe is important according to my research. I mention Scrum master, cycles, product backlog and other items that are important when developing with Scrum. In the next chapter, 5.2 I explain how the team in Oslo and Stavanger, and probably Abingdon even though I didn't observe them, work with the changes of Scrum, and how that has changed their way of work. In the next chapter after that I will write about some theories that is about Scrum and usability, and here is when I need to use references that is from the Internet that have not been thoroughly checked because it has not been done that much work on Scrum and usability because it is a relatively new research area. I used a reference from a blog where the author is supposedly a usability engineer. I have to accept this source as valid, but I treat my findings with caution. In chapter 6 I discuss some of the findings and results I got when interviewing and observing the teams in Oslo and Stavanger.

5.1 Scrum – definition

Scrum is a work process that is commonly used in software development. This is an iterative incremental framework which emphasizes close cooperation between members of the development team. Though the process may seem simple, it strongly affects the way software developers work and is made to be very agile and adaptive. It promotes that the

software teams direct themselves, without any project manager that decides when to do what.

When you work with Scrum, you work in cycles, often of a month, but it is possible to have it shorter or longer. The whole team decides when to do what, together with the product owner. You have also a product backlog which is basically a list of requirements or items that has to be done, with priorities on them to show how important that is, and if it is needed to do this in the next sprint or cycle. A Scrum team should be 7 or less, but it is also possible to let several Scrum teams form a project. (Larman, 2004)

5.1.1 Incremental method

Scrum is an IID method incremental and iterative development) and is a method to build and develop software. Each iteration of the Scrum is consisted with is a whole project, with requirements analysis, design, the programming itself and testing. After the iteration, you have an iteration release which is a tested part of the whole system. This is just internal; they don’t actually deliver something external to clients before the final release which is the final product to be delivered.

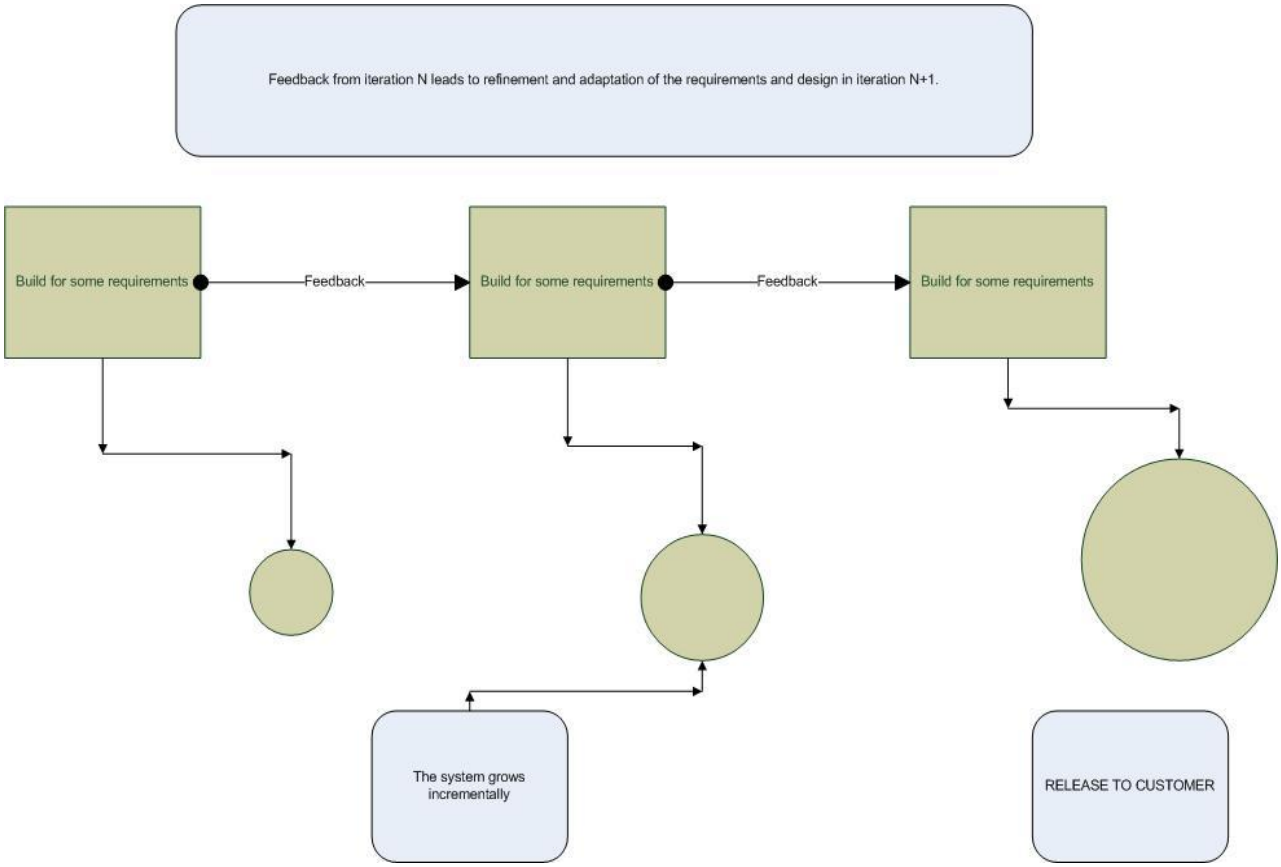

Figure from (Larman, 2004)

Scrum is a work method that fits both small projects and projects with many developers, this is because of its ability to form many small teams, and that certain representatives from each team meet each day. It is very adaptive, and can be used in many kinds of software development. It is also easily combined with other types of work process.

5.1.2 Lifecycle of Scrum

The lifecycle of Scrum has 4 elements: Planning, Staging, Development and Release. In these phases all the work is done, from starting the project to release the finished system. Here is an overview of the phases:

- Planning phase: In this phase the developers start out with deciding which requirements they can start, put up a budget for it, estimate time of development, usually in hours it will take for a developer to finish it.
- Staging phase: Is used to identify more requirements and put priorities on each of them.
- Development phase: Implement the system, planning of the upcoming sprint, defining the backlog, they have daily standup meetings and sprint reviews.
- Release: They deploy the finished system, prepare documentation and have training of the new users.

It is not possible to add more work during the sprint. A sprint usually lasts for one month at the most. If the case is that a task has to be added to the workload of the team, you should remove another task to make up for it. At the end of each sprint, there should be a review meeting where the team, the Product Owner and other stakeholders attend and it is shown a demo of the final product.

5.1.3 Roles in Scrum

When someone is developing using the framework SCRUM, you have four types of roles; the development which is called the Scrum team; that usually consist of the developers; you have a product owner, and management; a SCRUM master. In addition to these, you have a group called chickens which is not allowed to talk during the meeting, but can observe if they want.

The development team works on each sprint's backlog and does the tasks there that have not been done already. Inside the team, there is no other role than "team member", everyone has the same role here. They update the product backlog during the iteration, sometimes as often as daily due to progress.

The product owner is the one that presents the requirements to the team and helps them set priority to the different requirements. He is responsible for creating and prioritizing the

Product Backlog. He is the one that chooses the goals to accomplish during the next sprint, and he also reviews the system together with any stakeholders at the end.

The SCRUM master is nothing like the "old" project manager from earlier development processes. He is not just from the management; at least 50% of his job should be developer as a part of the Scrum team. This is a person which doesn't take decisions for the team, but rather encourage the team to decide for them self and also sees to that the developers gets as rare disturbances from the outside as possible. This helps the developers focus on their own work, and that they don't have to sit on the phone all day talking to clients. He ensures also that the Scrum is followed, and it is his responsibility to prepare for the daily standup meetings so that they go well. He is the link between management and the developers.

5.1.4 Daily standup meetings

The standup meetings are held at the same time and place every day during the development phase.

Each day of development, the team has something called stand – up meeting where they discuss:

- What have you done since the last Scrum?
- What will you do between now and the next Scrum?
- What is getting in the way of reaching your goals?
- Add any tasks to the Sprint Backlog?
- Have you learned or decided anything new?

This makes the daily standup meetings effective and a way for the developers to learn what the others do. It is called a standup meeting because the participants are supposed to stand during the meeting, which is one incentive to keep it short. The team uses a whiteboard

where they record which tasks that are done, in progress or pending. They use yellow sticky notes to write the task on so that it is easy to change it from “in progress” to “done”.

It is required during the meeting that if there are offsite team members, they should attend via teleconference or phone call. The meeting should always start on time, this is to keep the disturbance as low for the developers as possible. It is not allowed to discuss other things than the questions listed below. If the conversation digresses, it is up to the Scrum master to get them back on track. If there are other issues that cannot be discussed on the standup, it is possible to have another meeting afterwards with a subsection of the people attending the standup.

The Scrum daily standup meetings are good for a number of reasons; it informs the team members of the status of the project they are working on, since they don’t report to a manager as they did before. Everyone that wants can come and watch the daily standup meetings, but the people that are not a member of the team cannot speak, but this creates openness in the organization.

When a developer stands up in front of the whole team and speaks about what he or she is going to do until next meeting, they give a kind of social promise to do it, and so it is more likely that they will follow through with their tasks. The meeting also provides measuring of the progress as mentioned, but also possibilities to adapt their work according to this. This is very important, as the software development is very unpredictable.

The scrum meeting helps the developers put their tacit knowledge into words with the question 5 above. This ensures that the team members develop and learn, and they share this knowledge with the rest of the team. Shared language, values and practices helps the developers create a foundation for common understanding, and Scrum facilitates this with these daily meetings.

5.1.5 Scrum values

When developing using a work process like Scrum, it is a whole different mindset from other work processes. Here are some values that are special for Scrum and so the people using Scrum should think about when working.

- Commitment: This means that the team should commit to the goal of the iteration they set once it is defined. The team itself is given the tools and opportunity to decide for themselves how to approach the problem. The management should try as best as they can to remove issues that block the development, when the team reports it in the daily standup meeting. The product owner commit to keeping the backlog up to date with priorities, guide the choices of work tasks for the next iterations and to give feedback on the results of each iteration.
- Focus: The team should stay focused on their work tasks with a minimal amount of interruptions. It is the Scrum masters task to try and keep the interruptions on a minimum. In addition, all things that stand in the way for development, as blocking issues and adding work load to the team should be removed by the management.
- Openness: The backlog with all the priorities is open to anyone, and the daily Scrums provide openness, where people can come and listen and see the progression of the team.
- Respect: This means respect for each developer, and working as a team. It is important not to place blame on each other when a certain goal is not met. The team members should be respected for their strengths. Problems are solved in the groups if a team member has any issues.
- Courage: The management shows courage by trusting the team how to manage themselves. The team needs the courage to take responsibility for the work that has to be done themselves. (Larman, 2004)

5.1.6 Scrum advantages and weaknesses

As all work processes, Scrum has both advantages and weaknesses.

The advantages of Scrum are mainly the reason why Scrum has been adopted extensively throughout software projects these last years. Scrum presents a new mindset, but it is fairly simple to understand and to carry out. Most of the decisions belong now to the team, and they get to decide more. This is a big advantage, because the team doesn't feel that the work they do and how they do it is forced upon them. It is their choice on how to approach the work tasks and the problems arising.

Scrum encourages adaptive behavior both from the developers and the system being developed. This is good, because it can save time when it comes to correcting mistakes early, and then it won't affect all the code afterwards. If you develop using the waterfall model, if you forget one thing in the beginning, then all the code needs to be changed later on.

It also encourages customer guidance of the project, which is through the product owner, and this helps the customer get what he wants if the communication is good in both ways. It also promotes team building through the daily meetings where the developers get to listen to what the other does; it creates a better feeling of belonging to the team. It also enforces a lot of team communication which helps the developers get to know each other, who is expert on what field, and bonding between the members. It also fosters innovation and learning among the developers when they need to take responsibility of their tasks.

One weakness is that it is a very new way of thinking, and if you don't scrap all your old methods and follow the "Scrum rules" it is very easy to fail completely. It takes some time to get used to this way of thinking, people are used with a project manager that tells them what to do, and many people like this. It can feel more uncertain for a developer when using Scrum as a work process. (Chetan, 2008)

Scrum gives minimal guidance for other issues than how the team should be managed. The focus is on the process, not specific software requirements techniques. It does not give

guidance to how the documents should be structured and named. This can be an issue when you have a big project with many teams, and if all the teams name the documents differently, it will be difficult to find the right document that you are looking for.

5.2 Changes for the team regarding work process

As I started getting to know the team in Oslo in January 2009, they went from a work process called RUP, Rational Unified Process, which is also a process framework. I never got a chance to observe the team how they worked before, but some of my interest regarding usability is to see whether changing the work process can help the team develop for better usability. When observing and talking to the team, I asked the questions in an informal setting about the changes they had done to their work process, so that I got an impression on how they worked before so that I could compare that to now.

Since I didn't have the chance to observe the team before when they used RUP, it is difficult for me to evaluate thoroughly the changes, but in the interviews with the developers, I asked them how they felt about the changes and the differences in how they worked. In addition to the interviews, I used theory about Scrum to see how other researchers see Scrum regarding usability and if my findings correlate with those. I know that my findings have to be treated with caution because of that, but since I didn't start cooperating with Schlumberger before after the process change, there was not anything to do with that. I discovered some interesting findings which I have included in the discussion section.

In addition, since this is a pilot project, only the Ocean teams use Scrum, and it is difficult to assess the advantages when the other teams around does not use this work process. Some difficulties could be from start up problems, or that all the other teams' use different time measurements than the ones using Scrum. The other teams still go through requirements for months at a time, but the Ocean team splits the requirement up into parts so that they can finish them during the two weeks.

5.2.1 The changes for the Ocean team from how they worked before.

The Ocean teams in Oslo, Stavanger and Abingdon are adapting to use SCRUM as a development process. These are pilot projects, to see whether Schlumberger could have benefits of using Scrum as a work process for all their projects. The development team works in sprints, where at the beginning of each sprint, they decide which requirements they should work on during the sprint. The sprint lasts for two weeks at a time. In addition to meetings in the beginning of each sprint, they have daily meetings called "standup meetings". These usually last ten minutes and is a sum up of what the team members have worked with since the last meeting, problems they encountered and they want to discuss with the group.

Before, the team worked in three months with a requirement before they delivered it. After this, the requirement was sent to testing, and the team got feedback on which things they needed to fix or change that they missed. Now, the team delivers the requirement after a cycle on two weeks, so the feedback loop has been shortened a lot. This is an advantage, because the team is already into the code, it's not that long ago they developed it, so it will be easier and less extensive than before.

Before they didn't have as many meetings, and when they had those, they tended to be long, and took a lot of the developer's time. Now, they have many more meetings, but they keep them short, and don't get as "boring" as those before and in addition, the developers think it is all right with the meetings and it is easier to get them to go then.

The team in Oslo has outsourced some of the work to a team of three developers in Budapest. After Scrum started and they have daily meetings, one idea was to use web camera to get the developers in Budapest to be more included. As I understand, before Scrum, the team had their own meetings in Oslo, and the tasks that they didn't want to do themselves were just pushed over to the people in Budapest. The idea with web camera is not in Scrum as that, but as a result of more communication between team members. Scrum has a rule that offsite members should be present, e.g. on a phone conference. This

team chose to use web camera to contact the consultants. This is better because it is easier to clear out misunderstandings when the team can see each other on the camera and it is easier to explain them, because they can see whether the person at the other end understands or not. In addition, before, they worked separately for two or three months before they presented their requirement, and now it is easier to keep on the right track when having more meetings and get to see what the others are doing. This is also better now with knowledge sharing between Budapest and Oslo and if someone either places has experience in a field, it is easier to ask that person when you have to meet them every day, instead of maybe going to Oslo from Budapest which is more time consuming and costly.

This not only improves communication, but also the sense for the people from Budapest that they are a part of a team, and not only hired help. Some of the consultants from Budapest have been a part of the Ocean team longer than the people in the office in Norway. If I had more time, I would interview some of the developers from Budapest to see if measures like this helps. In addition this could lead to more sense of belonging and cooperation, and that will help the developers be more productive when they get the feeling the whole team is working together to accomplish a task, and that they don't just get sent work that others don't bother to do themselves.

Now the consultants from Budapest have a chance to say what they want to do themselves, and they can say no if they don't want to do it or don't have the expertise to do the requirement. This is good for the team that they can take discussions easier. Still, the team in Budapest has a certain area of expertise, so that the dividing of tasks often falls naturally.

Scrum has no mechanisms for providing usability and this can be a problem if the developer is not aware of it. Again, we are not talking end user usability, but usability for the API. The process is even more producer driven than before, since it is more regular checks on the progress, every other two weeks, and the goals they set is important to hold. It is probably easier for the team now to set more accurate goals, but if one developer uses extra time to fix a usability issue, it can affect the deadline and the whole team. But it is more agile than earlier work processes, so the changes to the code can be easily done even

after you have started developing. In addition, closer communication between the developers lowers the bar for asking for help. “The one I need stands right next to me; I can just ask him a quick question or maybe he can come and look at my code.” This also increases the awareness of the different expertise in the group, “Hey, do you know anything about this?” This makes it faster to get help, and increases the efficiency.

5.3 Scrum regarding usability

Scrum does not have any specific means to enhance the development for usability. Usability gets more and more important as argued in chapter 4 and in Scrum we have to ask ourselves “Where does user- centered design fit into Scrum”?

In addition, in Scrum there are small iterations, which do the testing difficult. (Chetan, 2008) It is possible to test the small part of the whole regarding functionality, but it makes it more difficult to test it regarding usability. Usability testing should be with test persons in a setting, where you observe the test persons do specific tasks and see where the test persons have trouble. This should be a part of the development when developing for good usability and Scrum makes this harder to do.

Scrum did not originate of the purpose for software engineering, and this can be one of the reasons it is not that good for usability. However, Scrum, as mentioned above has many other advantages over other work processes. In addition, it is possible to try and adapt user centered design into scrum, by trying to test as many possible usability issues as possible and then give feedback so that the corrections can be put into the next cycle of work.

Chapter 6 – Findings

Petrel is very big software that is used by oil companies all over the world. This software is already very popular and Schlumberger earns millions of dollars off of it. The process from the end user to the developer is long in this scenario, because there are two links, and this makes it difficult to maintain good communication between the end user and the developer of the API.

In this chapter I present my findings, in chapter 6.1 is the findings of the first question: "How does a developer understand what the user of the plug - in developer wants?"

In chapter 6.2, I present my findings regarding how the process going over to Scrum can affect the usability in the question, in the next section, I present a finding regarding something called an asset team, which could help with the usability and early user input in the development process.

In 6.3 I describe my finding regarding the Ocean developers and usability, what do they actually consider while developing, is usability important for them? Section 6.4 describe my finding that the team actually is very aware that the API should have good usability and uses time to make sure that it has it. Also here I describe the reason they mostly are concerned with the API usability, and not as much with the end user usability.

Chapter 6.5 describes some of the challenges that I found for the plug - in developer when he tries to make the plug - in have good usability, which will say that the plug - in has the same user interface and work flows as the original Petrel.

6.1 How does the developer understand what the user of another developer wants?

A requirement is created when the end user needs a new feature and gives the requirement to their developer, the plug - in developer. Then that plug - in developer sits down with the

Ocean product owner to create a requirement. The plug - in developer explains what he needs, and the needs of the end user are somehow implied in that, but the requirement does not contain explicitly what the end user need. Sometimes there can be misunderstandings due to this, because the plug - in developer just conveys the requirement he needs to fulfill the needs of his user. But if the Ocean developer also knows what the end user wants, it is easier for him to foresee misunderstandings for example. Sometimes also the Ocean developer knows easier ways to give the end user what he wants and not do it the way the plug - in developer wants.

A story one of the developers told me was that he often experienced that the plug - in developer wanted something, e.g. an algorithm to do a specific task. When the Ocean developer delivered the requirement the plug - in developer told him he needed, it was not what the plug - in developer actually needed, he just thought it was, and he could not use the algorithm to realize the end user's needs. Then the Ocean developer said to me that if he knew in advance what the end user wanted, he could easier foresee what the plug - in developer needs.

The Ocean developer also gets their requirement from the Twiki page. If there is something that he doesn't understand, he contacts the product owner that can clarify problems and contact the client on behalf of the developer.

"You would have to go back and forth for a while and try to understand, at least, what they want to achieve. In some cases it's obvious, some cases not." Developer 5.

So the Ocean developers use a lot of time trying to understand the requirements, so one possibility is to also include the end user's needs in the requirement, so if the Ocean developer don't know at once how to best solve the requirement, he can consult the end user requirement and use that to clear up misunderstandings. The Ocean developers have a lot of experience in Ocean, and know more about its capabilities than the plug - in developers, so he can solve the issues in another way than the requirement from the plug - in developer asked him to do. This double requirement can take some extra time to create, but they will save on extra time used to clarify what the clients actually needs.

6.2 Scrum

The changes in the work process have been quite extensive. Scrum requires a whole new way to think and work, and the team has been going through quite some changes.

6.2.1 Positive effects of Scrum

The communication between the team members have increased but in small intervals, which the developers like. Many of them have some aversion with everything that smells of work process, those with much experience claim that there is a new "revolution" every third year or so regarding work processes that is going to change the world. So the people rooting for Scrum has to do that carefully as to not get the other developers against the idea.

After Scrum was adopted, the team has more meetings, but they are shorter than, and not as extensive as earlier meetings. This is one of purposes of the standup meetings, the team has to stand while doing them, which is one effort to keep them short.

"We definitely have more meetings, but they are much more casual than what we traditionally thought of as meetings. So, our daily stand – ups they don't feel as much like a meeting because we don't go to a meeting room and set up a projector or call in on the phone line so we have more meetings, but they are not as painful. " Developer 3.

Before when they didn't use agile development, it was very cumbersome to do changes. If you changed a name, it could be that you had to change a lot of the rest of the code. Now it is supposed to make changes as you go along, and this helps that it is easy to change a name etc. The daily meetings are supposed to make the team up to date on changes of names, and they often use it to discuss. Still, even if it is possible to change the naming early on, there still have to be fixed other places in the code.

In addition, when using Scrum, one of the developers said that more people are watching the code and can point out faults in usability, or maybe have a better solution to a problem

you have been struggling with. Maybe he or she has struggled with the same problem and can give insight to the problem area.

"Given that the communication within each Scrum team is more open we have some benefits due to that because more people are looking at the code earlier."
Developer 1.

In addition, as they have the daily stand up meetings, they include their team in Budapest on web cam so that they can discuss things that they do and what problems they have etc. Before going over to Scrum, the team in Oslo took the parts that they wanted to do, and then delegated the rest to Budapest. This probably made the team in Budapest feel not part of the team, but now there are two – way communication, and the team is much better in cooperating now than before.

One developer mentioned that after going over to Scrum, they saved money in form of efficiency. Before, they started on more requirements, and if they finished three requirements 80% they lost the rest because their time was up, and they couldn't finish any of these, and a lot of the work was lost. But after they started Scrum with two weeks sprints instead, it is just one requirement at a time they finish, so if they have three requirements, they do they after one another, so that they will have two finished requirements, and one unfinished, and in that way, they don't lose as much work.

The amount of user input is the same as before going over to using Scrum as a work process, so regarding that there is no improvement for usability.

6.2.2 Negative effects from Scrum

The Ocean team is developing a framework that has existed for some time now. Due to that, the team has a lot of legacy code, and Scrum makes it harder to deal with that. When they now develop with more changes early in the development it is difficult to also think about the legacy code.

"We have a lot of legacy code and a lot of dependencies on Petrel and those don't really map well into the Scrum ideology" Developer 1.

"Because a lot of things that I think about when I try to make the API usable, it means having lots of reviews, having lots of people look at it, getting lots of feedback and because those things don't map well into Scrum tasks, it's difficult to plan for them, so I don't think we have as many informal reviews, I think a lot of people feel too pressured to get their assigned scrum work done and they don't have as much time just to casually look at other peoples work and I think in general it has become more end – product driven so the goal is to have something out the door and not necessarily spend a lot of time working on it and reworking it. " Developer 6.

This quote is taken from one of my interviews and this is extremely interesting. This means that even if the Scrum method is supposed to make people cooperate more and help each other on the team, this is not always the case.

It is more difficult for the commercialization team to test the small requirements both when it comes to functionality, but especially the usability. They get small pieces of software each other week that they test. But we see later, the research found that there are not many usability issues that the commercialization team runs into.

6.3 Ocean developers are mostly concerned with the usability of the Ocean API

What I learned when I interviewed the informants was that the Ocean for Petrel developers doesn't think that much of the usability of the end user, because it is so far ahead in the future for them. They rarely get to see the finished product, and the usability of the plug - in is so much up to the plug - in developer.

"*We have seen demos of what the end – user is getting, but we don't really see a lot of what it is going to look like, but then the usability of what gets delivered to the end – users are almost entirely in the hands of the plug – in developers, so the plug – in developers could do a really good job and make their plug - in look like it is a part of Petrel."* Developer 7

They first of all concentrate on the usability of the API, and then the end user usability comes in second place. This is logic and understandable, since the Ocean team has so little control over the end product. But they need to understand what the end users need in terms of functionality and, but the user interface is rarely or never settled upon before the start of the Ocean development. The developers try and make it easier for the plug - in developer to make the plug - in look like Petrel, but sometimes it is very hard, because Ocean doesn't have all the mechanisms that Petrel do. As I understand, the Ocean developers do as best as they can in the current conditions.

6.4 The Ocean team spends a lot of time discussing usability for the API user

The Ocean team and Schlumberger Information Solutions is generally aware of creating good usability for their users. Every one of the developer has their own definition of usability, because I included the question "What do you think about when I say the word "Usability""? On this question, I got a lot of different answers, but mostly it was that the API that they developed should have good usability and be easy to use for their users. Also, when asked about which criteria that would be the first thing to be excluded of a requirement if the team had to choose because they didn't have time to finish all, nobody said usability. I would be of the first four criteria, and this is extremely good, regarding other applications where the usability is often the last thing to be done, and most easily excluded.

"The first thing to go is the examples, the second thing to go is the documentation, the third thing to go is the actual API design itself in terms of usability," Developer 1.

This statement above was from one of the developers answering the question above on which criteria that would be excluded first. Below, I also include another quote regarding this:

"I would think usability is the first thing we think about. I mean especially if the functionality has a public part as you maybe know, our public part has to be very stable, we cannot change it from one release to another release so when we make something public it has to be useful and it has to work and be stable." Developer 3.

Usability is important for the team and they use a lot of time trying to make the usability as good and the API as easy to use as possible, considering the constraints of money and time.

"A large number of our discussions are, I mean usability based in the sense that we are trying to make choices based on usability to try to make it hard for the client to make mistakes, for example." Developer 3.

"As an API developer we need to think about usability for another developer so we strive to keep consistency and also try to simplify as much as possible." Developer 2.

When I interviewed the commercialization member of the team in Oslo, I got an impression that there rarely are usability issues that they have to comment on. This confirms the impression I have of the developers when interviewing them, that they uses a lot of time discussing usability issues for the API. Mainly naming of methods is the most common usability problem, and to keep the API consistent with earlier versions and also the other developers code. Schlumberger employees know very well what usability is, and for the Ocean developers, it has been emphasized so that the team knows its importance.

The developers in Schlumberger use a lot of their time to actually discuss usability issues. They rather use an extra half an hour to get it right so that the methods name is unambiguous and so that it won't confuse the user. But they mostly are concerned about the API user, not the end user of Petrel, they just want to know which methods they should expose to the API so that the plug - in developer gets his or hers functionality. The developer of ocean should think more of exposing functionality so that the plug - in

developer has the possibility to make an application that looks the most as the original, Petrel. This is the optimal here, that the user should not know the difference between Petrel and the application.

6.5 Plug - in developer usability

For the plug - in developer, the main issue here regarding usability is to make the plug - in be seamlessly integrated with the Petrel application. The ideal with this scenario is that the user shouldn't even know that he is not using Petrel, but a plug - in added to Petrel.

One of the usability problems Schlumberger experiences is that it is sometimes difficult for the plug - in developer to write their plug - in with good usability. Some things that are already easy to do in Petrel, is not possible to do using the Ocean Framework. This means that the user experience will be degraded because the plug - in will not look like Petrel at all. This becomes the problem of the plug - in developer that either has to do it themselves with a lot of complicated code, or he chooses to not try and make it look like Petrel at all, because it is simply not possible. In addition, the problem is when there are so many plug - in developers developing with their own style, that the user experience will be degraded, and sometimes it will have bad usability because the plug - in doesn't have the same user interface as the real Petrel.

Sometimes it is not possible to make the plug - in look like Petrel, like I mentioned above, and sometimes the plug - in developer does not care, and just makes the easy way to do the user interface with no regards to Petrel.

6.6 Examples and documentation

The finished API consists of the code, documentation and examples of use. Many of the developers I interviewed seems to think that more examples would help a lot on the usability, but they feel that it is not that much time to do that, and that it takes time away from their production time and their bonus. But if the developers used extra time to create

good examples it would make it easier to use for their users, and in that way save time, even if they go out of production for one week a year to make examples.

"To improve the usability of the API, we should gear our efforts more towards creating samples than we do now, I think. Because we do develop samples but is not very frequent and is not very systematic, some people do it, and some people don't. And I think that Ocean as a group to spend a little more time on it." Developer 2

This is an interesting finding, because this can increase the usability a lot, and they have a policy to do that. But as stated in the quote above, not all developers do it, and it is sometimes just some small examples. To make rules that the developer should spend that much time, or make this many examples would help the usability a lot.

*"Because we have people here in Oslo, people in Stavanger and in Abingdon and they tend to diverge a little bit in terms of style, in terms of naming and in terms of structure etc. If you know how to use the API's that are produced in Abingdon, you may not know how to use the one that comes from Stavanger. And I think that is a usability issue because it means that they have to learn the patterns for the API three times. "*Developer 4.

6.7 Summary

In this chapter I described six types of findings that were important during my research. I described how the developers got to understand the end users needs, and in the next chapter, I will suggest how to improve this to ensure that good usability is reached when developing a framework, both for the end user and the API user.

Also I described how Scrum affects the usability; in this case I would have needed more research from the way it was before using Scrum because I just got a chance to observe the team when they used Scrum, not before. So my findings here are based on observations and interviews with the team members.

I included descriptions on one issue that could help usability, to put together an asset team to make sure that the usability is ensured from the start. In the chapter after that, I

described how the developers think about usability and how much effort they actually use on the subject on usability. I also raises issues about the example code, how the plug - in developer can develop for good usability regarding Petrel, and that there is not that much user testing during implementation. In the next chapter I will discuss my findings and also come up with suggestions on how to improve some of the findings to make it easier to develop for better usability.

Chapter 7 – Discussion and analysis

I this chapter I will discuss my findings from chapter 6. I will discuss all the problems and then come up with possible solutions to some of them. In chapter 7.6 I analyze my theories, and in chapter 7.7 I reflect upon the work during my research.

Good usability ensures that an application is consistent, easy to learn, keep the error rate low, the speed of task execution should be high. (Schneiderman, Plaisant 2005).

When the Ocean API has good usability, many advantages will be seen from that. Below, I will list some of the benefits with good usability regarding this case.

When the time to learn the Ocean API is reduced, Schlumberger can use less money in teaching the clients. Both Schlumberger and the clients will benefit from this. In addition, when the API has good usability, it also ensures user satisfaction, and the clients would recommend the software to other companies and thus earn more money on the application.

When you keep the error rate in an application low, then the speed of the development is higher and the clients save money. In this setting, when I talk about clients, I mean both the in- house and the external clients, so Schlumberger will benefit from these advantages, because they also use the Ocean API.

Schlumberger have a lot of people around the world using Petrel, and as many of them works in shifts, e.g. 3 weeks on, 3 weeks off, this criteria is essential so that the employees doesn't need to learn Petrel all over again each time. The retention time should be low, as discussed in chapter 3, theory.

7.1 The users in the case

The fact that the end – user is not a person that is working with IT originally creates a special demand for good usability in Petrel. The usability is used here about Petrel when it looks as similar as possible to the original application, and the ideal should be that the end user has no idea he uses a plug - in instead of Petrel.

One important issue to point out here is that when the user of Petrel is not an IT educated person, the user of the Ocean API in most cases will be. This will mean that it is easier to develop something for another developer which thinks more like the person developing it. The difficulty will be the step from the API user to the end user, for him to really understand what the end user needs, and provide that, this means both functionality and good usability. But of course, the usability should be as best as possible even if they develop for another developer and it is easier to develop for someone with the same background as you, the communication is easier, and there is less room for misunderstandings.

This is probably one of the reason, that the Ocean for Petrel team doesn't receive as many usability question and feedback to fix it as many other applications. As the commercialization team said; they rarely have to ask them fix naming of a method, because both the team is aware of the dangers of bad naming and in addition the people using the API has an IT background themselves.

7.2 Some challenges and solutions to them

Here, in this section I will describe some of the central problems in the Ocean for Petrel development, and this is regarding both the users of Ocean and the end users. It is about the usability, and what means that can be used to ensure that is has good usability.

7.2.1 The long chain from end user to developer.

The process a requirement go through from it starts at the end user side until it reaches the Ocean developer is long. It is many people in between, and the more people involved talking to each other on different times, the room for misunderstandings increases. On solution to this can be to

gather the whole chain of people, from the end user, to the Ocean developer. This includes the end user, plug - in developer, the Ocean product owner and the Ocean developers. They can sit down to discuss the usability and the functionality, and also create a prototype on paper to try and foresee how the user interface of the plug - in in Petrel should look like. To make prototypes is one measure to ensure usability, so if they can agree on the user interface in the beginning this can be done this way. But as earlier stated, the user interface is very much up to the plug - in developer himself. This is of course a very costly way to do this, but can prove itself to be one way to do it in some cases. It can help the Ocean developer to see the final result, to make him know which methods need to be created and how to proceed to make that happen. The product owner is also one solution to help the process, and will be discussed later.

7.2.2 Asset teams

In Petrel, they set together a team for each project or new requirements, which is called an asset team. This is a good idea to do also for Ocean, and to get the commercialization team earlier in the loop. The asset team should consist of the Ocean developers, product owner and the commercialization team. They can give feedback to the team before they start working and thus prevent the team into doing things wrong. The commercialization team has a lot of experience in testing the software, and then can check to see if some of the common mistakes are being done. In Scrum this can be easier to do. So when a new requirement comes in, the product owner, the developers, the usability architect for Ocean and one or two from the commercialization could form a team to discuss how the work should be done to ensure that all aspects of the requirement is taken care of. This ensures the criteria earlier of user input in the development phase which I talked about in chapter 3.

7.2.3 User input

In Ocean, they don’t have as much user input during the implementation. One of the usability issues is that the usability of the API is not tested enough during the

implementation, just at the end. From chapter 3 we see that user input during implementation, and also testing is an important agent to ensure good usability. Schlumberger has a multiple choice questionnaire which they present to new users to measure the usability. But this is mostly too late, because then they have already implemented the framework, and the changes cannot be done for another two years. A solution to this can be to start testing the functionality and the usability earlier in the process. Scrum presents a challenge regarding this, but when there should be testing by end users during the implementation phase so that the developers can use the feedback to improve the design.

The problem with usability regarding the stability promise for Ocean happens when a developer does something wrong, e.g. names a method ambiguously such that the API user takes the wrong one of two. If there are 500 plug - in developers, and 50% uses the wrong method, or uses a long time to figure out which to use, 250 developers are wasting a lot of time. This is not really smart economically. And in addition, they have to keep this for two years. If they have user input and feedback from them, they the chance of doing mistakes, and the usability is stable but also good.

7.2.4 Consistency

The main methods to keep the API's usability good, was from the developers point of view consistency and also consistency of naming, which will ensure all the usability attributes mentioned above. To improve the consistency it is important that the developers communicate and agrees on what style to use.

In API development, the most usability problems are of naming of methods. The naming should be clear and not ambiguous so that the developer or the persons using that method are clear on what it does from the documentation and naming.

Ocean has promised stability, which means that they cannot change the API before two years. This is done because the code their users create should not have to be updated all the time, they

should be given good time to fix their changes before the method becomes obsolete or some other changes. This stability ensures consistency, which is one of the criteria for usability, and one of the more important ones when it comes to API usability.

The developers developing Ocean has their different styles when they program. You can probably see almost which programmer that wrote which part of the code etc. This makes the usability and the consistency difficult to maintain. Even if people followed the same naming and programming principles there will always be room for interpretation and misunderstandings. In addition, if two developers discuss how they want to do a method or something else, they can think they agree when they don't. In addition, one developer can choose to say that he agrees with some other, and then end up doing it his own way anyway. And there is always room for misunderstandings; the code isn't clear until it is written down. Sometimes it would be easier if they just decided to use one type of developing because the users can be confused by the different styles. If they work on one part of the API where there is a special way of naming things, and then move on to use another part of it, where the naming is completely different, the user needs to learn a whole new set of thinking and programming style.

One solution to this would be to make a set of programming and naming rules that everybody should follow, in Oslo, Abingdon and Stavanger. This will help the consistency, and prevent the different parts of the API looking different due to different developers.

The plug - in developers outside of Schlumberger is more difficult to control that their application looks like Petrel, but the in-house developers and the Schlumberger plug - in developers is easier to control. But as there are a lot of Schlumberger plug - in developers in Beijing and other countries that are situated far away, and they usability engineers are sitting here in Norway, there are bound to be differences in style.

One problem with the API developers for Schlumberger is that they are over in China and lack of appropriate training in how the GUI should look like and guidelines makes their plug - ins not look like Petrel at all. This degrades the usability and functionality in Petrel because if the workflows and GUI doesn't look the same, the user's needs to learn a new way of thinking in those parts that are plug - ins, and a lot of the advantage of plug - ins goes away.

There are some rules that the plug - in developers follows, but they should have a class teaching them how to do it and also why the usability and the consistency in Petrel are important. In addition, there should be some set of rules in the cases that there is not possible to make the GUI look like Petrel so the consistency still is there, even if the user interface does not look the same as Petrel.

7.3 Scrum

In my findings about Scrum, I have two quotes from two different developers. One of them says that Scrum makes it easier to do changes in the code and use feedback from the others to make changes, and the other one actually claims that it is worse. One interesting thing about this is that these two developers are from different teams in Oslo and in Stavanger; this could be a result of different practices. Also, the one that claims that the communication and feedback was better, he said that maybe because that is the way Scrum is supposed to be. If I include the theory on the subject, Scrum should help with the feedback and the communication. One thing is that since it is still early in the changeover to Scrum, eventual problems can be due to the fact that it is a new way of thinking and that the team has not gotten quite used to the way of working.

Some of the theories to develop for good usability are to have a lot of user input. As I interviewed the developers I asked what they thought about changing to Scrum and if they thought that would help for the usability. Most of them argued that it didn't have anything to say, because they didn't get more user input now than before. Some argued that it was more difficult since the team was more about producing a lot of code, and if they used extra time discussing a usability issue, it could lead to the team missing its deadline, and not finish the requirements they set out to do. Most of them agreed that since they worked more closely together, it was easier to discuss certain usability issues. In addition to this, since Scrum is an iterative framework it was easier to go back and change something if the usability was not good enough, because Scrum is made to be easier to change the requirements or code according to new input. In addition to all over the above mentioned things, the team has switched from working on one requirement for three months to be able to deliver something every two weeks. This makes it easier to change thing, because you don't sit and work on a requirement for three months, then get feedback, and

then have many things to fix at once, it is easier when you can get feedback from other developers and fix it as you go along.

Many of the problems the team encounter with Scrum is probably due to difficulties in the start, and does not directly affect the usability in their development. Some of them find it difficult with the two week deadline, which the managers only use time to measure the product, and not quality. It can maybe be easier for the developers to skip the usability part if they are on a tight schedule, because Scrum does not provide any mechanisms to prevent that.

The team has gotten more delivering products oriented, they have to produce code faster than before, and this can be on the cost of e.g. usability as is claimed by some of the developers.

One issue when it comes to using Scrum as a work process is the time limit. Before, when the team actually got to finish a whole requirement in three months or more than one before they left it to testing and the commercialization group, they delivered a finished product. To be able to deliver something now in two weeks, they have to break down the requirement into pieces, and then it is more difficult to test the functionality for the testers. Since the team is a pilot project in Scrum, and no one else is using this method inside of Schlumberger Information Solutions, the Ocean team is working at another pace as the rest of the organization. This creates disagreements sometimes with the other teams, because before when people asked the developers to fix things, they could take a day or a couple of hours to do it, but now, they have to wait until the next sprint so that the developers can schedule them in.

As long as only one group is doing Scrum it will lead to differences like this, but that is just in the beginning before the people involved gets used to the concept of delivering things every other week. Some of the Scrum problems have just something to do with startup problems, and will decrease. But as mentioned, Scrum will not directly help with the usability, the team doesn't get more user input than before, but it can be easier to fix the usability and get corrected by others during development than before, and such as a byproduct it can help with the usability.

But still, the Ocean commercialization team, works following the system engineering method waterfall model. This means that the Ocean development team is developing a requirement or a

part of that finished, and then delivers it to the commercialization team for testing. One idea is to close the loop tighter between the developers and the commercialization team. This has happened more now with Scrum, because before, they worked on a requirement for three months before delivering it for testing. Now at least the intervals are shorter and easy to give feedback. This again makes it easier to make changes and provide for better usability.

7.4 The product owner

Some of the problems in this scenario are due to the fact that it is two links from the end user and the Ocean developer. Sometimes it could be good to meet with all these together to try and form the requirements together. The middle, the plug - in developer, the Ocean developer and the end user and in addition the architect for Petrel could be a part of it. It could be better to try and make prototypes of the final result and the GUI so that all is clear on what to expect. This could help the Ocean developer see what the result is supposed to look like so that he can make it easier for the plug - in developer to achieve the goals.

This middle link can seem cumbersome, why can't the developer contact the clients directly, but this is mostly because the clients are both from Schlumberger and from other companies. Sometimes if the client is from Schlumberger the developer takes contact with him themselves, maybe they know them after cooperating for many years or other things. In addition, the product owner shields the developers from being called up all the time with requests and "Hey, could you just fix this for me, it will only take a minute". If the product owner didn't have this role, the developer would be swamped with these kinds of requests and would not have time to work at the developing at all.

7.5 Usability

In API development, the most usability problems are of naming of methods. The naming should be clear and not ambiguous so that the developer or the persons using that method are clear on what it does from the documentation and naming.

The Ocean for Petrel developers tries to think about the end users, but they have also their own users to think about. If they want to expose a part of the API to the user, they have to be careful so that it can be used by as many as possible. If they make something extensive and advanced for one client, that the rest of clients find confusing or don't have the need for that function, they have to make a decision whether they should create the requirement or not. If it makes it worse for all the other clients, it's just not worth it. Stability and that it is usable for most people possible is the most important goal, it is not possible to satisfy every client.

Ideally it should be as much user input as possible, both in the beginning of a requirement, but also during the development process, and they should provide testing etc. The problem and difficulties in this case, is that you have two links before you reach the end user, see figure from chapter 3.

"The developer would typically make a plug - in which then the end user would use in Petrel, so we need to know or have a fair idea of what they are trying to achieve, because we must know what data they need to make their plug - in."

The problem with just trying to make the new applications look like the old Petrel can be that it can be difficult to learn by new users, if the initial design wasn't good enough, something it rarely is. Software needs to develop during time and feedback from the clients, therefore is it not always the best to just hang on to the way it was made from the beginning. This will always be a trade - off between existing users and new users, the usability can be better, but make it worse to use for those already used to it. but you need to think about the new users also, because in the long run, the cost for training for the new users will be very large if the software is difficult to use. I tried the software and found it difficult, but that is probably because I don't have any background in geophysics or geologists and therefore didn't get to use it to something useful, just to play with it to see how it worked.

7.6 Analysis

If the usability could be the only criteria we could go for, you have to have a lot of money and time to do it thoroughly. Most companies does not have endless time or money, they are developing something that they want to sell to others, and then other things that usability comes in mind. To make a profit, they have to be efficient, and deliver the requirements on time as promised the customer. These are extra requirements in addition to usability that helps keep the customers happy.

It costs a lot of time and money to have people giving feedback. You have to take both the developers and the users out of their regular jobs so they are not in production, and in addition maybe fly them half across the world with costs in hotels and flight fares.

The creation of the asset team is in this case somewhat difficult because most of the commercialization team is situated in Bangalore. There is possibilities for video conferences, which would save a lot of money in travel and time for the involved parties.

Since the effects of good usability will not show themselves before after a year or more and then in total, for example the total cost from a requirement starts at the developers for Ocean until the customer has a finished window or more possible features in his program. It probably will show that just if the Ocean developer used one extra day to fix some requirement or a method, then 500 developers wouldn't do as many mistakes, and then they would save a lot of money in both time not wasted and extra cash, because the requirement makes it faster to the market. This leads again to satisfied customers which will continue to buy the product. And in many years, we will see the expenses reduced in training costs and use when it is easier to learn and use the software. This cost is not as easy to measure, as the cost running the Ocean team, and each link has to think about them and to keep the costs as low as possible. The reason this is like this is of course logical, the other cost is impossible to measure, and the developers think that if they use longer time than what was agreed on in the beginning, they lose their bonus. If they make something that is difficult to use, and their users is not very productive because they do a lot of mistakes regarding usability and uses longer time than necessary, which is taken from their bonus.

There are some features in Petrel that are simply not possible to "copy" for the plug - in developer and the ocean for Petrel developer. This makes it a little bit worse, because even if the developer tries to make the usability good, the stability and consistency cannot hold because of that. Since one of the best usability principles Schlumberger holds to is consistency this has a lot to do with usability. When a user is used to Petrel working in a certain way, they will get uncertain if they open a plug - in which doesn't look like Petrel at all. It has at least to behave as Petrel, even if some things look different. The buttons and the choices the user has should be similar to Petrel.

In addition, if the team has a usability problem, and they choose to skip it because of time pressure, then their users could end up using a lot of extra time trying to figure out how to do things, and in total, Schlumberger or another company ends up losing money. So in total, Schlumberger would save money and time to let the developers of Ocean use a little bit extra time on the usability, so it would save more time for the other chains. One of the reasons that the developers are not using enough time on the small things that could matter a lot is the bonus system.

"I mean, if you think of Schlumberger as a whole, spending one more week fixing a property is nothing. Because you're going to save it up easily with those 500 developers that will not be confused. But if you look at your project and your project only, you actually save time. But that's not how you should think, but how you get rewarded." Developer 3.

They get measured on time, and not quality, but they work to get the quality as good as possible between these constraints. One of them argue that if they don't finish a requirement on time, it ruins their bonus but if the others uses a long time to figure out something it is on their bonus, so that it doesn't matter to them because they only get measured on time and not quality. One way to fix this problem is to change the bonus system of the developers, so that it measures other things than time. But since those other qualities are difficult to measure, it is difficult to deal out bonuses according to it also.

7.7 Reflections

It was a challenging task for me to get to know how Petrel and Ocean works, not just because they use a different programming language than I have a background in, but also because this is an interdisciplinary field which combines IT, programming and geology, petro technical science. Some of the developers in SIS are pure computer science, while others have background in geophysics or geology. This is a good thing to have people know both sides to make the end product as good as possible. In addition, the team in Oslo hired a new guy that have been a user of Ocean for many years, and he have a lot of competence in the usability of Ocean and what works and what doesn't work as well. I used a lot of the time in the beginning to get used to the language and all the new words that I didn't understand.

I got to use both the end product and the API which gave me the feeling of a regular user. Sometimes I encountered problems, and because of that it was easier for me to put myself in the user's shoes.

As long as I understand that my presence could have colored my results, and that another researcher may have gotten another result, despite the fact that I tried in every possible way to minimize the factors, you can never take a researcher totally out of the research. This is because even if I am not always aware of the fact, my gender, my education, age, where I am from will color my results. My age is because I feel sometimes insecure about it, because I am so much younger than the people that I have studied in this case. Gender is because it is mostly men who are working there, but they also have women which are good, but mostly men as developers. My educational background from computer science will also help me have e.g. more compassion and understanding of the difficult environment of the developers, than I would have had otherwise.

References in this thesis: To explain the most basic things in my thesis, I have used relevant literature from books or journals that I can accept as guidelines from experienced authors. However, I was in this case forced to use some references from the Internet which I have to use with caution. Some of the references is from Schlumberger, and I had to use the Internet, because there aren't any books written about this exact case. I accept these references from the

Schlumberger website to be correct and up to date. One other issue about my field of study is that there are not so many books or literature written on the subject, because this case is kind of unique. In addition, when it comes to Scrum, which just recently became one of the buzzwords and really well known to companies, it has not been written much literature about the effects of Scrum, regarding usability or other effects. I found a book regarding Scrum which describes it, but not about the effects it can cause. That is why I have been forced to use some references or sources from the Internet which I cannot guarantee the validity for, and therefore I have to treat my results with some kind of extra caution.

The reason for me writing it in English is that Schlumberger is an international company with people from all over the world, and so that the people at Schlumberger can get something out of my thesis. In addition is most of the work already done in this field in English, and I have experience in writing in English and I felt that it is more professional to write the thesis in English. The literature that I reference to is almost only in English, so that made some of my work easier. Sometimes it is difficult to translate words from English to Norwegian because I feel that we have fewer words to use, and I was afraid that some of the meaning was lost.

Some difficulties can appear since I have not English as my native tongue, e.g. that I have misunderstood some of the meaning of what people said or didn't get the whole context because neither I or my interviewee have English as a native tongue. To try and reduce the possibility for misunderstandings, I sent my transcription of my interviews to my interview objects so that they could see through it and correct me if I made any mistakes or assumptions that were incorrect. I didn't get any feedback from my interviewees so I have to assume that I don't have made any big mistakes regarding misunderstandings.

This was one of the issues I encountered during my work: Some of the weakness of qualitative research is that the researcher gathers so much data; it can be very overwhelming to start to analyze all the data. This can lead to a loss of overview over the subject when the researcher delves deep into details discussed in interviews.

Chapter 8 Conclusion and further research

Usability is and will always be important for companies producing software. It will help their clients do their job more effectively, and that keeps the clients happy. To do that, Schlumberger has hired me to see into how things are now, and if there are things that can be improved from the way they work now. Petrel is already very successful software, so they must be doing a lot of things correct already, but it is always possible to improve them to still be the leading software in this field. I come from outside with no connection or anything to Schlumberger, so that I can look at these problems with fresh eyes.

8.1 Conclusion

Schlumberger already has a product they earn a lot of money on. There are always things that can be better, but sometimes you have to weigh the cost of time and money up against the improvement of usability or new features. Still, in this thesis I discuss some improvements that is possible to do something about the feedback, to increase the user feedback earlier on in the development process, and also to involve the testers which is the commercialization team more during development than just testing the requirement after it is finished. After all, the commercialization team has a lot of experience not only about the usability, but also about the bugs and other issues that they run into, and that can be avoided from the start when including them earlier on.

As a result of Scrum the rate of communication increases between the team members, but even so, this does not mean that the usability automatically increases. The user input stays the same as before Scrum, and Scrum as a working method makes developing for usability harder, despite the fact that the communication increases. The communication helps the usability, because there are more developers looking at the code, and that in turn makes the others aware of the work they are doing and can provide consistency easier than before. The amount of user input stays the same as before the team started using Scrum.

I have also looked at how the developer for an API understands the requirements, not only in the light of their users but also the end users of the Petrel application. This is a complicated process, which in chapter 7 I proposed some ideas to improve this process.

8.2 Further research

Because this research has been quite limited, both in duration and extensiveness there was some topics during my research that it could be interesting to know more about, and if I had more time, these are the things I would like to look more into.

The things I didn't look into is how the part of the team situated in Abingdon worked, or how the support team could give feedback on usability, because they know and see where the users have the most problems. This is interesting topics that I wish I had time to research.

To improve usability, it is possible to use the feedback given from the clients when they ask for help. Schlumberger has a support team, and I would have liked to interview one of these to see if they know which usability issues that are most common, and make rules that would avoid usability mistakes like that. Schlumberger has all this feedback and they don't take advantage of it, it would be easier and less expensive to use information like that, than to gather all the users and have them answer a questionnaire or a survey.

It was also scheduled for me to go and observe another team in Abingdon, but I decided that I didn't have enough time this time around, but if I had more time, I would go to Abingdon to check for similarities and other ideas. Perhaps the developers there would have another way of doing Scrum that would have helped the usability, or perhaps they had other ways to develop and discuss the usability.

In addition I would like to check out more about the consultants from Budapest to see if the process of going over to Scrum really is positive for them regarding increased cooperation with the Ocean team in Oslo.

I would also have interviewed an end user of Petrel that requested a plug - in to see what became of that, maybe follow the process from a requirement gets started, and until the end product. This can take a long time, up to a year, so I would have to have a lot more time available, but would be extremely interesting to actually see what happens from a requirement is made until the plug - in is ready to use, and if the usability of the plug - in is good and similar to the Petrel software.

References

Argyris, C. og D.A. Schön: *Organizational learning II.* New York: Addison-Wesley;1996

Chetan, D. (2008). Usability Innovations for Scrum. [Online]. Available from

< http://blogs.oracle.com/usableapps//2008/01/usability_innovations_for_scru.html>
[Downloaded 15. June 2009]

Dalland, O. *Metode og oppgaveskriving for studenter.* 2nd edition. Oslo: Gyldendal Akademisk; 2000

Koelle, D (2007) *API Usability.* [Online]. Available from: <http://www.davekoelle.com/api_usability.html>
[Downloaded 25. May 2009]

Larman, C. *Agile and iterative development – A Manager's Guide.* Boston: Addison – Wesley; 2003

Mathieu J (2004). *API Usability: Guidelines to improve your code ease of use. Free source and programming help.* [Online]. Available from: <http://www.codeproject.com/KB/usability/APIUsabilityArticle.aspx>
[Downloaded 23. June 2009]

Myers G.J, Badgett T, Thomas T.M and Sandler C. *The art of software testing.* P 135 – 137. 2nd edition. New Jersey: John Wiley & Sons, Inc; 2004

Nielsen, J. *Usability 101. Definition and fundamentals – What, why, how (Jacob Nielsens Alertbox)* [Online]. Available from <http://www.useit.com/alertbox/20030825.html> [Downloaded 28.June 2009]

Oates, B.J. *Researching Information Systems and Computing.* London: Sage Publications Ltd; 2006

Robson, C. *Real world research, a resource for Social Scientists and Practitioner - Researchers. 2*nd edition. Oxford: Blackwell Publishers Ltd; 2002

1. Schlumberger (2009) *About Schlumberger*. [Online]. Available from <http://www.slb.com/content/about/index.asp>
   [Downloaded 15. June 2009]

2. Schlumberger (2009) *Schlumberger Information Solutions (SIS).* [Online]. Available from <http://www.slb.com/content/services/index_sis.asp>
   [Downloaded 5. June 2009]

3. Schlumberger (2009). *The power of Petrel 2009, Schlumberger*. [Online]. Available from <http://www.slb.com/content/services/software/geo/petrel/petrel2009.asp>
   [Downloaded 22. June 2009]

4. Schlumberger (2009) *Plug - ins for Petrel, Schlumberger*. [Online]. Available from <http://www.slb.com/content/services/software/geo/petrel/petrel_plugins.asp?>
   [Downloaded 3. June 2009]

5. Schlumberger (2009) *Ocean, Schlumberger*. [Online] Available from <http://www.slb.com/content/services/software/geo/petrel/ocean_apidk.asp?>
   [Downloaded 3. June 2009]

Schneiderman B. and Plaisant C. *Designing the user interface.* 4th edition. Boston: Addison - Wesley;2005

Thagaard, T *Systematikk og innlevelse, en innføring i kvalitativ metode*, 2nd edition. Fagbokforlaget Vigmostad og Bjørke AS; 2006