Methods for developing visualizations in a non-designer environment

A case study

Vera Antonov
Adam Sterner

Tutor: Rita Kovordanyi
Examiner: Jalal Maleki
Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Vera Antonov, Adam Sterner
Methods for developing visualizations in a non-designer environment: A case study

Adam Sterner
Linköping University
Linköping, Sweden
adast736@student.liu.se

Vera Antonov
Linköping University
Linköping, Sweden
veran441@student.liu.se

ABSTRACT
Teams consisting of only software developers will occasionally need to develop products that has to be easy to use. User Centered Design (UCD) is one approach to help increase the ease of use of a product and that can be incorporated into the teams’ traditional workflow when needed. The software developer team followed in this thesis had not tried to incorporate UCD into their current workflow when developing such products. So, this thesis looks at how a product designed with an agile/UCD approach differs from a product developed by the software development team’s traditional approach. The two products were designed to solve the same problem. The results show that the product developed with an agile/UCD-approach gave better usability rankings and better external appeal than the one created with the team’s traditional approach. On the other hand, traditional methods constitute a better choice for quick development of products for more technical user groups within the company for example.

INTRODUCTION
This thesis has been written at Ericsson AB in Linköping. A cross functional team consisting of software developers who develop machine learning algorithms for radio networks was followed for this thesis.

This team spends most of its time developing and improving machine learning algorithms and analyzing the performance of these. The work within the team is conducted using an agile method.

A lot of the work that this team does is software development as a part of a bigger product. Occasionally the team faces a situation where they must create internal solutions that demand usability of a certain level. In these cases, an adaption in method is seldom made as usability and design are not part of the team’s standard development procedure. When the solution’s value is dependent on usability and design then using the team’s traditional method might make the solution’s value suffer.

Traditionally these cases are handled in the same approach as other tasks within a sprint. The task is delegated to members of the software development team who will develop a solution to the best of their ability.

Since the team has its focus on products that do not require an extensive usability it is not feasible to recruit a user experience (UX) designer. It is an extra cost for the team and a designer might not fit in with the rest of the team’s workflow. Therefore, it is of interest to see what the effects of incorporating approaches of User Centered Design (UCD) into the work of the current team are. This would let members of the team take on the designer role in a UCD approach. Which leads to the following research question.

RQ: What is the difference in measured usability of a visualization of data built in the teams’ traditional software development manner compared to a visualization built in a combined agile/UCD approach?

An introduction to the specific data set used can be found in Background.

Delimitations
This research question will be explored within the contexts of a single case of visualization of data and by developers with little to no UX-design education. Visualizations are interesting since they are intended for end users to gain further understanding of the data, an understanding that might not be visible when just looking at the data in the present use case. Therefore, the users’ understanding of the visualization is interesting to look at when measuring differences in methods.

1 Radio networks are networks of radio towers, base stations
**BACKGROUND**

Development is done to achieve faster network connections and better connectivity by looking for improvements at all fronts.

Many computing devices use the radio network infrastructure, and this is where some improvements could take place. One point of improvement can be the hand-overs from one frequency or radio tower to another. Hand-overs can be compared to the search for a new channel on a car radio when the signal to the current channel is getting too weak to listen comfortably to. This switch to a better channel is called a hand-over when dealing with connections to a radio network and is performed when a device gets bad reception and therefore changes to another frequency.

Reducing the search-time for a new frequency during a hand-over is one way of improving the connectivity in the network. The team at Ericsson whom we have worked with during this thesis uses machine learning as an approach to achieve this. Their machine learning models use the frequency the device was at, its radio position at the time and which frequency it switched to learn to predict which new frequency should have a good connection, resulting in better connectivity and less search time for the device since not every single frequency needs to be tested by itself as is done today.

The models in the network are continuously validated and retrained with new data from their surroundings to make sure they perform at the required standard. This makes the model go through different states during its lifecycle.

It is of interest for the developers of these machine learning models to be able to follow the lifecycles of the models in the network in combination with other data about the model. This can be done by simply trying to read the data, but it can be a challenging task with such a high complexity. Visualization as an approach for better understanding might be necessary.

**THEORY**

The following sections will give insights to the different approaches of developing products such as the one to be developed in this thesis.

**Agile**

Agile software development is a collective name for an iterative, agile method of how software development should be done. The Agile Manifesto [1] stating how agile development should be conducted was created in 2001. Agile methods focus on delivering working software frequently with a preference for short intervals, often called **Sprints**.

---

2 Radio position for a unit is the signal strength for the neighboring radio cells at the same frequency
According to Sy [12] sprint 0 can include the following activities:

- Interviews/Conducting contextual inquiry
- Gathering data to refine product/release goals. Making sure everyone in the team understands them.
- High-level exploratory designs to derive design principles.

Interviews are according to Wilson [13] either structured, semi-structured or unstructured. Structured interviews being a verbal questionnaire, unstructured interviews a conversation with a user/stakeholder about a topic without any predefined questions and semi-structured interviews being a combination of both. Semi-structured interviews are seen as most fitting in our method over a structured or unstructured approach. Wilson recommended semi-structured when gathering data about a topic which is not completely unknown to the interviewers and to understand user goals and workflows. All of which fit our situation where some information had been presented to us beforehand by the team.

Interviews alone are not enough to understand the work and should be complemented by observing the users do their work in their natural setting [14]. This is a description close to the definition of contextual inquiry, which is described by Bayer and Holtzblatt [15, p. 41] as:

*The core premise of Contextual Inquiry is very simple: go where the customer works, observe the customer as he or she works, and talk to the customer about the work.*

When the data from interviews and contextual inquiries have been compiled this can serve as basis for high-level designs like prototypes. The high-level exploratory designs are used to derive the design principles for the rest of the project. To boost creativity, techniques such as creative workshops can be used. Maiden and Robertson [16] uses a two-day workshop consisting of four half-day sessions, each based upon different creativity techniques to find new ways of thinking. The resulting designs can be tested on users using prototypes and the data gathered will guide design decisions during the rest of the project.

One way of prototyping designs is to use a “throw-away-prototype” [17]. When using a throw-away-prototype one must be careful not to end up developing the main product on top of the throw-away-prototype. France and Rumpe [18] means that valuable lessons can be learned from doing so but that finishing the end-product may not be possible.

*Sprint n*

In the following sprints after Sprint 0 it is discussed that there are two distinct roles: designers and developers.

During Sprint n the designer(s) will be testing the functionality of the developed features in Sprint n-1 as well as designing for Sprint n+1. The goal for the designer is to deliver a specification of a design combined with tests that can be used by the designer to evaluate the implementation once implemented. This design specification shall be grounded in the design goals and principles.

To create designs, prototypes can be used. Often low-detail prototypes [12] [19] such as paper prototypes are used early on in a design-phase to quickly evaluate different designs. It is argued that it is favorable to create two or more early prototypes to show a potential user to gather feedback [20]. This is to eliminate the “nice-bias” of the person presented with only one design not wanting to hurt the designer’s feelings. If two or more designs are presented and comparison can be done, people tend to have an easier time discussing the design [20]. The input on the low-detail prototypes can serve as ground for development or can be used to create high-detail prototypes that can include interactivity and flows.

**Usability evaluation**

To measure the usability of the products developed within each method a method for the evaluation is needed.

<table>
<thead>
<tr>
<th>Letter grade</th>
<th>Numerical score range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>84.1–100</td>
</tr>
<tr>
<td>A</td>
<td>80.8–84.0</td>
</tr>
<tr>
<td>A-</td>
<td>78.9–80.7</td>
</tr>
<tr>
<td>B+</td>
<td>77.2–78.8</td>
</tr>
<tr>
<td>B</td>
<td>74.1–77.1</td>
</tr>
<tr>
<td>B-</td>
<td>72.6–74.0</td>
</tr>
<tr>
<td>C+</td>
<td>71.1–72.5</td>
</tr>
<tr>
<td>C</td>
<td>65.0–71.0</td>
</tr>
<tr>
<td>C-</td>
<td>62.7–64.9</td>
</tr>
<tr>
<td>D</td>
<td>51.7–62.6</td>
</tr>
<tr>
<td>F</td>
<td>0–51.6</td>
</tr>
</tbody>
</table>

Table 1 – A translation from SUS scores to letter grades

Regarding the concerns about whether the method suggested by Silva da Silva can be converted into practice it was deemed manageable because the team creating the new application were only two people and had similar knowledge within both design and development.

*Sprint 0*

In this section the details about how to execute sprint 0 will be reviewed.
A common method for evaluating a system is using so called task completion. The test person is given a set of non-trivial tasks and the test leader will then observe if the person completes the task or not. This can be evaluated as if the user has not succeeded within a predefined time, the task is considered a failure. Task completion can also be measured by the number and level of “prompts” the test leader must give the user for them to succeed [20].

Task completion is a good and common measure of usability in a system. But it has a disadvantage of being specified in measuring the usability for a single system. When comparing two systems’ usability using task completion it can be hard to create tasks that will not be bias towards one design or the other. This is why we will not use task completion in as an evaluation method in our thesis.

Usability is how appropriate something is for pursuing a purpose [21]. This means that every system needs to be assessed within its environment. Different systems need different methods. This makes it time consuming and not very effective to measure usability and hard to compare usability between different systems. In 1986 James Brook developed a universal scoring system scale called the System Usability Scale (SUS) [21]. The test consists of ten questions which are answered on a scale 1-5. The score is then calculated using a predefined model which will yield a SUS-score of 0-100. According Sauro [22] a score of 68 is considered average based upon analytics of more than 5000 user scorings since the SUS score of 68 is ranked in the 50th percentile. From this Sauro and Lewis [23] present a translation of SUS scores to letter grades that can be found in Table 1. The SUS evaluation method is now considered an industry standard and is widely used.

Visualization of data

UCD-methods such as interviews or user tests can be used to find what variables makes sense to present together. The greater number of dimensions the greater the complexity of the visualization will get. Iliinsky presents that the most common number of dimensions are kept under three to four but that there are successful visualizations of six, seven or more dimensions [24, p. 3]. Composing different types of data visualization is a way to get more variables on to the same visualization [25].

There are many tools for visualizing data. High-level tools like Microsoft Excel can be used to quickly get charts and graphs for a data set but the possibility to personalize and interact with the visualizations is limited. If the information is interactive, it is easier to get that “aha-effect” from the recipient [26]. Lower-level tools like the R libraries ggplot and Shiny can be used to create powerful visualized experiences but the time needed to create the visualization can be expected to increase compared to higher-level tools.

When designing the data visualizations, it is important to know the usages of different visualizations. Grinstein, Wierse and Fayyad [27] discuss different usages for a visualization system such as if it will be used for exploration of data, confirming a hypothesis about the data or finding good views to be able to present data in a good way. Which main usage of the system has been found during research will also determine the type of implementation which will be best suited.

Previous work

A lot of studies have been conducted within the field of Agile/UX methodology to see what obstacles can arise using it and what tools are best to be used with it. We have found very few, if any, studies that address the actual output of programs developed in this way.

The few who have investigated it have not been very well recognized and their validity can therefore be questioned.

A study conducted by Sensuse et al. [28] examines a system developed in a UCD/Scrumban approach using SUS. The findings are not compared to any other system.

A comparison is conducted by Schwartz [29] where an examination of whether the quality of software improves if a design expert is present within the team. Schwartz compares a team where a member of the team has the designer role and a team that has a trained designer and concludes that the SUS-score improves when a trained expert is present.

Adikari et al [30] is also conducting research comparing two teams. One team works in a traditional agile manner and the other with the help of a UCD-designer. They could show using SUS that the product developed by the team that had a UCD-designer on the team scored better than the other team. Worth noting is that the products developed both scored well under the average measure for a SUS score which is 68, by scores of 47 and 53 respectively.

This shows that some research has been done within the area but that there is room for further investigations on the topic. Especially regarding using UCD methods without taking in a UCD expert into the team.

METHOD

This section will present the development process of each of the two approaches separately. Then the testing of the approaches will be explained.

Traditional approach

The traditional approach was conducted by a pair of two members of the original developing team. They were assigned the task of developing the visualization during one of the team’s sprints. One member drew a sketch of bar-charts on a piece of paper and then they wrote R-scripts3 to manipulate csv-logfiles into visualizations as the

3 https://www.r-project.org/
Agile/UCD approach
The agile/UCD approach was carried out following the high-level framework suggested by Silva da Silva [2].

Sprint 0
Sprint 0 was conducted according to Sy’s [12] recommendations to gain a deeper understanding of the user.

Semi-structured interviews were carried out in the manner presented by Wilson [13]. The interviews were held with four, senior, practicing software developers which had earlier been identified within the project scope as our primary user by the company. Each person was called to a 30-minute interview via email and was interviewed separately. The interviews were held all in one day, in the same, closed, conference room with at least 30 minutes between each interview. The spoken language was Swedish. The questions used in the interview can be found in Appendix A.

Contextual inquiry was conducted at the end of each interview by asking the user to bring his/her laptop to the interview. The user was asked to show us an analysis they use often and walk us through the steps of creating the data/visualization and how they read the output.

The results served as basis for defining the user goals for the system. These were defined either as qualitative goals (general and non-quantifiable) or quantitative goals (objective and measurable) [19]. When the goal gathering was finalized we followed the guidelines of Comstock and Duane [31] by checking the final user goals with relevant managers before development began.

The goals serve as basis for coming up with two completely different design tracks which the team got to vote which track to follow. These designs served the purpose of exploring two distinctively different design ideas early on to determine which design guidelines to follow during the rest of the project. The design tracks were also presented to the users in written form to gather feedback to make a choice about which design track to follow.

The chosen design track according to the user feedback served as basis for the beginning of development in sprint 1. Specifications were made on what base functionality needed to be implemented in the form of user stories.

The developer role had nothing to do in this sprint and since it is extensive in work for the designer-side the involved people worked as designers during this sprint. The sprint was also two weeks long in comparison to the following sprints being one week each.

Sprint 1
The developer’s task was to develop the base functionality for the upcoming system according to the specification and user stories from Sprint 0. The development was conducted on Linux using the programming language R with a library for building interactive web-apps called Shiny4.

The designer’s work during sprint 1 was to design for the next sprint. The designer developed user stories from the design goals and two high-level prototypes in the form of paper-prototypes. When finished the prototypes were shown to two different users asking them to compare the prototypes to gather their thoughts on the design. The chosen prototype was then used to do design choices and user stories for the next sprint.

Sprint 2
The developer developed the functionality specified for the sprint. The designer did user tests on what was developed the previous sprint. The results from the user tests were used to create user stories and prototypes for the next sprint.

In addition to paper prototypes the designer created high-detail prototypes using Microsoft PowerPoint resulting in a specification for the developer.

Sprint 3 and further sprints
The designer uses the tests which were developed for the design of the last sprint to test the design and implementation on two users. Feedback from the tests were provided to the developer. The designer also began work on the design for the next sprint in the same manner as described for Sprint 2. Instead of doing paper prototypes the high-detail prototype from the last sprint was now updated with the new design for this sprint.

The developer developed the new functionality specified for the sprint and incorporated the feedback from the testing on what was developed in the last sprint.

Finalizing
A total of 4 sprints was performed and the result after testing and implementing feedback on the last sprint was the final product. This product was then tested using the quantitative goals as a quality assurance.

Usability evaluation
The test documentation can be found in Appendix B.

Tests were conducted with four members of the team who had not been consulted during the development of the Agile/UCD approach. The two systems to test was the developed Agile/UCD product and the earlier existing traditional product used for analyzing the life cycle data. Only one of the two authors were present during the tests to not create a threatening environment.

---

4 https://shiny.rstudio.com/
The test person was at first presented with a small background of the project, that it would be a test of the systems and not of them as individuals and that all data would remain anonymous.

They were then presented to one of the systems on a computer and were handed a task sheet where the person was asked to browse the system and see if they could find any analytical findings. They were given 7 minutes to do so. Then they were asked to rate the level of how interesting they thought their findings were. The author present was not to help the test person during the tests.

Directly after the completion of the tasks the users were given the SUS-questionnaire and were asked to score their experience without thinking too much about the questions.

This process was then repeated with the other system with the same test person. Two of the test persons were presented with the Agile/UCD product first and the other two were presented with the product developed in the traditional manner.

Lastly a single question was asked to six people outside the team concerning their interest in the visualization. A scale of 1(not interesting) - 5(very interesting) was used. Three of them were shown the traditional product, and three of them the Agile/UCD product. After being shown a product they answered the question. Both products were shown on a computer screen.

RESULT

This section will present the results of the work.

Interviews

The current team described themselves as a team where there are no fixed roles other than a rotating scrum-master role. Some informal roles are mentioned such as managing the communication with operators which deliver data. The team analyzes data from collections which are gathered a couple of months in-between. When new data shall be analyzed, a pair within the team takes on the task of creating visualizations of the data. This is done through formatting the incoming data, running it through R-scripts and finally visualizing the data using LaTeX reports. These reports can then be compiled by other members of the team to view. All code is shared through the version control system git. The analyses have been used for understanding the models better as well as for quality assurance. In some cases, the analysis has led to finding bugs in the code among other things.

This same method was described by all the interview participants. One participant described another complex tool called ITK which has complex functionality to explore data with multiple dimensions.

According to the participants the best thing about the current workflow is that it is easy to reproduce findings and that they do not have to rely on third-party software. The negative is that it consists of quite a few manual steps to produce the report and that the results are not visually impressive.

Today they usually construct the report or look at the visualization in a broad analysis but with a specific question in mind in comparison to prior deep, detailed analysis.

All the participants had good knowledge about the concept of lifecycles. During the interviews it was clear that there is a hope to conduct broader, more exploratory analysis of the data and gaining new insights. Potential insights to be found are described as where a model stays in the same state for multiple cycles in a row or “flickering” between states. These behaviors of a model are described as especially interesting in combination with for example geographical location or time of day.

It is also mentioned by most of the participants that it could be nice to have more visually appealing graphs as well as material for easier explanations of the data for those not very familiar to machine learning.

Traditional approach

The resulting product was a pdf-report consisting of 477 pages. It included two tables consisting of all nodes in a radio network and a counter of how many times a model associated with said node stayed in the same state as in the previous measurement (0, 1, 2, >=3). One table showed the results for the state “OUTDATED” and the other for “NOT_OUTDATED”. The rest of the report had one page per model. Each of these pages showed state transits in a bar chart (Figure 2). It was estimated that this took each member 8 hours to complete which gives a total of 16 hours development (Figure 4).

Agile/UCD approach

Sprint 0 was the largest sprint conducted. From the information gathered during the interviews, a set of user goals were summarized. Goals were for example “The system shall be used to analyze data to see patterns/deviations” or “The system shall be easy to use even when used infrequently”. The goals can be found in Appendix C.
From the goals and the interviews three potential design tracks were presented to the group:

1. A static report like the one used for analysis today
2. A semi-interactive report with drill-down functionality
3. An interactive application

Most of the facts gathered from the interviews pointed us towards an interactive approach. But when presented with options, many of the participants commented that they would prefer one of the report-options. This because it was closer to the current work context and could be more useful in their daily work.

This showed a need for a product that can work in their daily life. But as stated by Unger and Chandler [20] it is important to not let the users stand in for designers since they will most often lean towards something familiar. This might stand in the way of innovation and new ideas. Therefore, we chose to go with the design track “Interactive application” since it is a product which will fulfill more of the needs of the team. Such as bigger support for multiple dimensions of data and more interesting presentations. But to keep the everyday aspect in mind, the application was chosen to be developed in R for the ability for the team to reuse the code in reports such as the ones used today by the team.

The following sprints are described below with a short summary of the main decisions made in each sprint.

Sprint 1 - Development of basic application with a map and a timeline. Design choices for sprint 2 were to use base stations as nodes on the map that were color coded for the majority state that they are in. Also adding user criterions and flags on nodes upon reached criterions.

Sprint 2 - Design choices for sprint 3: Mouse over-events for information about the node. Separation of criterion for flagging and timeline. The flagging will use a time interval instead of a timeline.

Sprint 3 - Design choices and tasks for sprint 4: To be able to add timestamps to the time line to easily go back to a specific time. Updating backend of how criterions in a time interval is processed.

Sprint 4 – Implementation of design choices from sprint 3.

The finished product was a web application developed in R using the package Shiny. It had an interactive map showing colored dots where the nodes were located, and these would change color when updating the time on a timeline. There was also functionality to save interesting timestamps (the red markings on the timeline in Figure 3) as well as flagging nodes based upon user input.

The total time spent over the five sprints in the Agile/UCD approach was 128 hours. These were divided on 49 hours of preparation, 27 hours on design and 52 hours of development (Figure 4).

Usability evaluation

The usability evaluation results can be found in Table 2, Figure 4 and Figure 5.

The evaluation showed that there was no great difference with regards to level of interesting findings in the two products. Both products scored an average of 3.

The SUS scoring (Figure 5) showed that the Agile/UCD approach reached a SUS scoring of 64.4 which according to Table 1 translates to a C- and the traditional approach reached a scoring of 33.1, which translates to a F.

In the external questionnaire regarding how interesting the visualization is to an external person the agile/UCD product scored an average of 3.6 with a median of 4. The traditional product scored an average of 2.3 with a median of 3.

Table 2 – The results of the user tests

<table>
<thead>
<tr>
<th>Participant</th>
<th>Agile/UCD</th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUS</td>
<td>Insight score</td>
<td>SUS</td>
</tr>
<tr>
<td>P1</td>
<td>70</td>
<td>3</td>
</tr>
<tr>
<td>P2</td>
<td>65</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td>P4</td>
<td>82.5</td>
<td>2</td>
</tr>
<tr>
<td>Average</td>
<td>64.4</td>
<td>3</td>
</tr>
</tbody>
</table>
DISCUSSION

Results
The results show that an agile/UCD approach yields a better usability of the system in comparison to the traditional approach. This is in line with the results from the previous work within the area. But it must still be stated that neither of the results can be considered good results with regards to SUS. Low scores were also seen in the previous work by Adikari et al [30]. This might just be a coincidence or proving the point that it is hard to develop well-scoring products. Adikari’s results had a smaller margin than ours’, our traditional product scored worse than their traditional product and our agile/UCD-developed product scored better than their agile/UCD-developed product.

It is also shown that the level of interesting findings rated by the users is equal between the two systems. This indicates that the agile/UCD approach and the added hours might not bring any remarkable new insights to the data presented. This is a bit surprising with regards to the theory which points towards that an iterative approach could involve more dimensions to data and therefore new insights.

From the external view the people interviewed were more interested in the agile/UCD product. This was expected since a UCD approach can be expected to have put more thought into the users’ perspective and therefore appeal more to the users.

Method
The most significant threat to validity in this thesis is the amount of work spent on each of the two approaches. The big difference in the number of hours spent makes it hard to conclude whether the results is due to the method or the higher number of hours that have gone in to the development.

The scientific reliability of the work is not high. This is because a lot of the work according to the method depends on creativity, and creativity is not very replicable. Therefore, the likelihood that someone following the same method would end up with the same results is not very high. The results with regards to usability could be the same though, that an agile/UCD approach is scoring better than a traditional approach. This has been seen in related works [30] and could most probably be seen again. To gain the same SUS scores would be much harder, if not impossible.

Since a lot of the research and testing was done within the team that developed the traditional approach, one can question the validity of the tests and scorings from the ones interviewed. They were already familiar with the traditional product and can therefore be assumed to already have interesting insights from earlier discussions within the team. This could lead to a higher score of interesting findings than if both products would have been totally new.

When testing prototypes or products, it is important to try to minimize the bias from the test person. A common bias is the so called “nice-bias” [20] where the test person’s will to be polite influences how they answer the test questions. To combat this, we chose to present our prototypes in pairs so that the test person could compare designs and feel more comfortable in giving us their honest opinions. But in other cases, one wants to eliminate the comparison, for example during usability evaluation. This was addressed by letting the different usability testers switch which product they started with, inspired by Adikari et al [30]. During the external testing one also wants to eliminate the comparison effect and here we chose to only show one product to each person.

Regarding criticism of the sources we have chosen to read a lot of research connected to the subject. The ones chosen to be presented in this thesis are the ones that were highly cited by others and work conducted by well-known names within...
the field. The sources that are of least reliability are the previous work within this field. There are few previous works that have been conducted and the validity of the ones we have chosen to present here can therefore be questioned.

This thesis in a larger context

In a bigger context we can see very few ethical risks working with either of the methods we have mentioned.

What can become a problem might be the stress it can cause a person to take a role, in this case designer, which one is not trained for. This might cause insecurity and stress if not handled right by the team leader.

It is also important to keep the ethical perspective in mind when conducting tests. Being a test subject can be a vulnerable position to be in and it is important to handle the identification of the test person with care.

CONCLUSION

The purpose of this thesis was to investigate two different approaches, methods, to create visualizations within a team of purely software developers. What the results show in this case is that an agile/UCD approach indeed yielded a better usability than a traditional approach.

In this case study, we could not see any difference in the number of interesting analytical findings from the two different approaches.

Our results in this case study indicate that using an agile/UCD approach is suitable when developing a product that needs high usability and to be externally appealing. On the other hand, traditional methods might constitute a better choice for quick development of products for more technical user groups within the company for example.

Future work

In terms of future work, it would be interesting to see a more extensive study on comparisons between traditional development methods and other different approaches involving UCD. It would also be interesting to study the effect that more sprints would have on the SUS-score.

REFERENCES


## Activity | Comments/Questions | Estimated time
---|---|---
**Introduction** | Welcome
  Introducing us
  - Our names
  - We're doing our bachelor's thesis
  - One of us is the interviewer, the other takes notes
  Explain the purpose of the interview
  - To better understand how the team works with analysis today
  Explain the interview method
  - First some questions then let them show us their typical analysis process
  - Estimated time 30 min
  - Anonymous data
  - Will compile the answers to find design goals | 3 min
**Structured topics** | T1: Interview subject background
  - What is your background and role?
T2: Current work process
  - How do you work with analysis of data?
    - What can the purpose/goal be?
    - Are your analyses usually exploratory or based on predetermined issues?
  - What tasks do you need to perform to do an analysis?
    - What tools do you use?
    - Can you describe the analysis process?
T3: Desired work process
  - What is the best part of the workflow you use for analysis today?
  - What could be better?
  - Dream scenario?
    - Follow up for specific contexts: Nightmare scenario?
T4: Lifecycles
  - Short introduction to lifecycles
    - Be sure to phrase it as one idea we have for a project to minimize bias.
    - Show the picture of lifecycles another team member has showed us.
    - Is this something you have heard of before?
  - Do you work with analyses of the lifecycle data?
  - What value do you see in analyzing the model’s lifecycles?
    - How did you reach these conclusions?
  - What other relations in the lifecycle data do you think would be interesting to find? | 15 min
**General questions/open dialogue** | - More to add?
- Contextual Inquiry: Ask them to show us an example of their analysis process on their computer | 10 min
**Closing comments** | Additions?
Thank the subject | 2 min
APPENDIX B

User tests

Task 1
Play around a bit in the system and try to see if you can reach any analytical findings that could relate to your team’s work. You will have 7 minutes for this, we will tell you when the time is up.

Task 2
Rate your findings. How interesting were they? Circle the number that most closely corresponds to your rating. It is important to note that it is your own findings you are rating, not the system.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Could not find any</td>
<td>Not interesting</td>
<td>Very interesting</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

External tests

As an initial reaction: How interesting would you find this visualization if it was presented in a meeting or conference you attended?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not interesting</td>
<td>Very interesting</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C

Qualitative goals – guides the design
The system shall be used to analyzing data to see patterns/abnormalities. For example:
- Events (states/transfers for example “OUTDATED” and “ON_HOLD”
- Fluctuations over time (weeks, days)
- Fluctuations geographically
- Skewness
- Frequencies/bandwidth-relationships

To go from a thought about an analysis to having a finished analysis shall be a relatively short process (few mouse clicks, commands etc.)

The system shall be able to be used infrequently and therefore have a short process to get accustomed to it.

Unnecessary information shall be hidden from the user

An earlier discovery shall easily be able to be recreated later on.

The visualization shall be appealing to enlighten interest within an external audience.

Quantitative goals - measurable
Measure time and number of wrong clicks for the following tasks:
- Find all models that have been outdated at least three times in a row

Users shall rate how easy it was to use the tool as a 4 (very easy) as an average on a 1-5 scale where 1 is “Not at all easy” and 5 is “Extremely easy”.

New users who have not used the system before shall rate how easy it was to learn the tool as a 4 (very easy) as an average on a 1-5 scale where 1 is “Not at all easy” and 5 is “Extremely easy”.

External people outside the team shall rate visualizations from the system as a 4 (very interesting) as an average on a 1-5 scale where 1 is “Not at all interesting” and 5 is “Extremely interesting”.

13