Automated User Interaction Analysis for Workflow-Based Web Portals

Emil Backlund[1], Mikael Bolle[2], Matthias Tichy[3] and Helena Holmström Olsson[4] and Jan Bosch[3]

[1] ATEA and Chalmers University of Technology
emil.backlund@atea.com
[2] Chalmers University of Technology, Sweden
mikael@bolle.se
[3] Chalmers | University of Gothenburg, Sweden
matthias.tichy@cse.gu.se|jan.bosch@chalmers.se
[4] Malmö University, Sweden
helena.holmstrom.olsson@mah.se

Abstract. Success in the software market requires constant improvement of the software. These improvements however have to directly align with the needs of the users of the software. A recent trend in software engineering is to collect post-deployment data about how users use a software system. We report in this paper about a case study with an industrial partner in which (1) we identified which data has to be collected for a web-based portal system, (2) implemented the data collection, and (3) performed an experiment comparing the collected data with answers of the test subjects in a survey.

Key words: user interaction, post-deployment data collection, Build-Measure-Learn, data-driven software engineering

1 Introduction

Customer satisfaction is key for a software product to be successful. To achieve this, software development companies need to know what their customers value and how they interact with the product [1, 2]. Without this knowledge, requirements prioritization becomes a challenging process in that product management has no accurate understanding of customer needs. In many companies, the feedback loop is slow and there are no efficient mechanisms that allow for continuous collection and analysis of customer data. If using "Lean Startup" terminology, there is no "Build-Measure-Learn" (BML) loop in place for continous validation of customer value [3]. In this loop, customer data serves as input for product management as well as for the entire development organization [3], and the feedback loop from customers is fast. Without the opportunity to continuously validate what customers value, there is the risk of lack of alignment between product and customer needs, as well as the risk of investing in R&D efforts without having an accurate way of continuously validating whether these efforts correspond to customer needs. What companies need are mechanisms that allow for continuous

learning about customers, about product usage and about what functionality adds value to customers.

Today, this knowledge is typically gained by interacting with customers using techniques such as use cases, scenarios, prototypes, interviews. Also, alpha- and beta testing, observations, and customer surveys are efficient mechanisms for continuously validating that the software functionality that is developed is of value to the customers. During the early 2000's, agile software development methods gained traction in most software development companies as a way to increase customer and end-user interaction [4]. With an emphasis on close customer collaboration and small development teams, agile methods have proven successful for shortening feedback loops and reducing time-consuming coordination processes for a wide range of companies [5]. However, and experienced as problematic, most agile techniques for customer and end-user involvement assume face-to face interaction between developers and customers, something that is difficult to achieve in an increasingly global and distributed development environment. In most large-scale software development companies there is limited, or no, direct contact with customers or end-users, and it might require costly and complex procedures to gather and extract in-depth knowledge about how a specific product, or a specific feature, is used. Moreover, asking customers what they want is problematic. As recognized in research within human-computer interaction decades ago [6, 7], the expected use of a system does not necessarily correspond to the actual use of that system. Often, there is a gap between what people say and what they do [8], which makes asking customers what they want a difficult task.

More recently, and due to the increase in connectivity and the on-line nature of products and systems, data can be collected as soon as customers use these. In Web 2.0 systems, and in on-line Software-as-a-Service (SaaS) technologies, automatic collection of customer data is the main source of input for learning about customers [9]. The cost of collecting data is low [10], and customer value can be validated on a continuous basis. Already, there is research indicating that automated data collection is conducive to an increased understanding of customer behavior and preferences [1, 2].

However, while automated collection of customer data allows for an increased understanding of customer behavior – what is it we learn? Does what we learn about a product by automatically collecting data correlate with the actual customer perceptions of that product? And what is it that we learn from automatically collected data that we do not learn by asking customers?

In this paper, we explore how to automate the collection of customer data, and how analysis of this data helps companies increase their understanding of customers. In particular, we are interested in comparing different data sources, i.e., what companies learn from different data sources such as (1) asking customers how they perceive a system, and (2) automatically collecting data about how customers use a system. In our study, we investigate how automatically collected data compares with results gained in a customer survey, i.e., how data collected by automatically measuring product usage compares with test results

when asking customers how they perceive the product. Our research questions are the following:

1. RQ1: What knowledge about how customers interact with a system is interesting for developers, testers, project managers of a web-based portal software?
2. RQ2: How well do automated data collection and analysis methods compare with actual user perceptions?

The case for our research is a software solution, called Accelerator, developed by ATEA Global Services that automates the tasks of a Service Desk. The software enables employees to perform IT related tasks like ordering software and hardware. Each task resembles a workflow which is divided into a sequence of steps which need to be performed to finish the task. The goal of ATEA Global Services is to automate as many of these tasks as possible. To achieve this, accessibility and understanding of end-user needs is essential. Therefore, ATEA Global Services wants to understand how these users interact with their software. The product that ATEA Global Services provides consists of thousands of lines of code, which makes it difficult and time consuming for the developers to implement a monitoring solution. On the other hand, a survey can also be difficult as they do not have direct contact with their end-users.

The contribution of this paper twofold. First, we present a list of knowledge needs about how user interact with a workflow based web system. Second, we developed an automated data collection method and compared the results in a user experiment with the users feedback in an online survey. As a result, we got a weak correlation between the answers of the users in the web survey and the results of the automated data collection.

In the next section, we discuss in more detail the needs for a valid automated user behavior analysis as part of the Build-Measure-Learn loop to enable companies consistently evaluate and improve their systems. Section 3 describes our research method to answer the presented research questions. The answer to research question 1 were identified by a workshop with developers, testers, and project managers at ATEA Global Services. The workshop and its results are described in Section 4. Section 5 presents the automated approach to collect user interactions to satisfy the knowledge needs identified before. In Section 6, we present the experiment to answer research question 2 how well the automated approach compares with an online survey with users. After a discussion of related work in Section 7, we conclude and give an outlook on future work.

2 Background

To increase customer satisfaction and to add customer value is key to any software development company in order to stay competitive [11]. Typically, ideas for improvement and innovation of features are collected and prioritized during the early phases of road mapping and requirements engineering, and as part of a

planned product release cycle [12]. The selection of what ideas to include is done by product management and forms the basis for enormous R&D investments. What has shown problematic is how to continuously confirm the correctness of the decisions taken during the requirements prioritization process. Often, product management has no accurate way to continuously validate whether the features they prioritize are those that add value to customers [1, 2]. As a result, requirements prioritization becomes a challenging process in which companies run the risk of having opinions inform decision-making rather than data reflecting actual customer usage. Moreover, and as recognized in human-computer interaction (HCI) research decades ago, asking customers what they want is difficult since what customers say does not necessarily reflect what they do [6]. In using well-established manageral behavioral terminology, espoused behavior is often different from the theory-in-use [8], meaning that there is a difference between what people say and what they do. To address this challenge, and to learn from customer usage rather than from their opinions, companies need to find mechnisms that allow them to continuously collect customer data. Also, they need to find ways in which to analyze and understand this data in order to understand what they learn from different data sources.

Below, we outline the Build-Measure-Learn (BML) loop which is a central concept within the lean startup community [3]. The concept is relevant in that it emphazises the importance of customer feedback, as well as the continuous need for this. Furthermore, we introduce the concept of data-driven software engineering. In relation to our research, both these are good examples of practices where continuous collection of customer data steer adjustments and decision-making. Also, these practices reflect the sincere interest in the software engineering field to find mechanisms that help validation of success in terms of customer value.

2.1 The Build-Measure-Learn Loop

As a central concept within the lean startup community, there is the "Build-Measure-Learn" (BML) loop, which is described as the concept of validated learning [3]. In this loop, ideas are quickly turned into testable products, data is gathered by measuring how the product is actually used by a selected group of customers, and ideas for product improvement and innovation are based on what is learned by analyzing the data collected from customers. In this way, focus is always on developing and delivering customer value, and the model advocates an approach in which continuous customer validation is critical. The concept was developed by Ries [3] after noticing that the solution focused thinking that characterizes the agile development approaches often leads to a situation in which many software companies fail. What he found was that instead of continuously evaluating what customers value, most companies spend time and money developing products without knowing whether customers will be interested or not. While projects are delivered on time and on budget, there is the risk that nobody wants the product. In the BML loop, the main intention is to emphasize the importance of continuous validation with customers in order to understand the problem during product development and improvement.

Besides the BML loop, another central concept within the lean startup community is the "pivot", a term used when a company changes direction based on what they learn from customers, i.e., from customer data. Ries [3] claims that having "pivoted" is the most frequently occurring commonality among successful startups, and that successful companies seldom end up doing what they initially set out to do. Rather, they change direction based on efficient collection and analysis of customer data and what they learn from customers.

2.2 Data-Driven Software Engineering

Data-driven software engineering is a practice in which continuous collection of data is used to understand the successful development of software systems [13]. During the development cycle, different metrics related to product quality are collected, and the goal is to use such metrics to make estimates of post-release failures early in the software development cycle, as well as during the implementation and testing phases. Such estimates can help focus testing, code and design reviews and affordably guide corrective actions and decision-making activities.

One area in which data-driven software engineering has been successfully applied is the area of Test-Driven Development (TDD). Test-driven development [14] is an "opportunistic" software development practice that has recently re-emerged as a critical enabling practice of agile software development methodologies after having been used sporadically for decades [14, 15]. With this practice, a software engineer cycles minute-by-minute between writing failing unit tests and writing implementation code to pass those tests. By using the data-driven software engineering approach, previous studies have investigated whether test-driven development actually work, and if so, if there is supporting data for development teams to make informed decisions during product development [14].

In summary, and in relation to this research, the BML loop and the concept of data-driven software engineering are both good examples of practices where continuous collection of customer data work as a basis for product development and improvement. These practices reflect the interest in the software engineering field to find mechanisms that help automatic collection of metrics, and as a result, continuous validation of customer value.

However, while there are numerous mechanisms for automatic collection of customer data, we are interested in exploring how well this data also compares with the perception held by customers. In our study, we investigate if there is a link between raw data and the assessment made by customers when asking them how they perceive a system. In the following sections, we outline our research method and the results from our case study including the results from a comparison between automated data collection results and customer survey results.

3 Method

In the following, we describe our research methodology as sequence of six steps as shown in Figure 1: (1) The understanding of the case and its challenges, (2) conducting a workshop to understand what should be analyzed captured as questions about user interactions, (3) breaking down the outcome from the workshop into a set of questions answering research question RQ1, (4) implementing an automated data collection for the system, (5) performing user tests including a web survey to generate data, and (6) analyzing the results from step four and five to answer research question RQ2. The fourth and fifth step ran in parallel because of their mutual dependency. Each step is discussed further beneath.

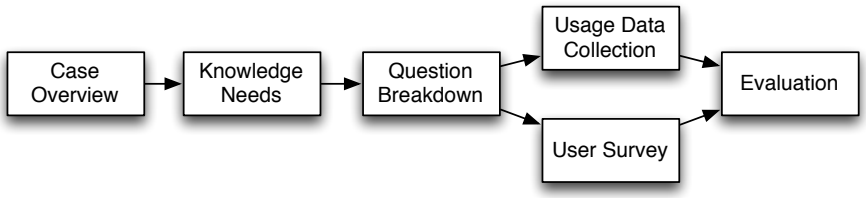

Fig. 1. Research method

To understand the case a brief introduction of the system was held by ATEA Global Services. Also, access to the system and the source code was provided for in depth analysis. Beside this, there were both formal and informal meetings conducted with different employees within the organization. Based on this information insight and general understanding of the case was gained.

Then, a workshop was conducted with employees of different positions at ATEA Global Services. The main goal of the workshop was to gain more insight of their perception of the system as well as identifying needs for knowledge about user interactions. The needs were formulated as questions about how users interact with the system.

Since the list of questions where not detailed enough for an automated data collection, they were broken down into more detail and made unambiguous, see Section 4.2. At this stage, it was also determined which questions could be answered by the automated data collection respectively by conducting a web survey.

To answer research question RQ2, we let test subjects perform a set of tasks and then let the them answer questions in a web survey, this is annotated in Figure 1 as User Survey. The tasks performed were related to a set of questions extracted from the workshop. The survey, see Section 6, was designed by examining the extracted questions from the workshop and their breakdown. Only a subset of the questions could be evaluated since some questions required that the data collection has been actively used for some time. With the questions for

the survey selected, tasks were designed that would be sufficient to answer those questions by the automated data collection. When the tasks had been designed the testing was executed by inviting users with similar profile as possible end-users. They were given an introduction of the case product, before performing the tasks, and afterward they were given access to a web survey. By having this approach it was possible to simulate a real world scenario. As previously mentioned, data collection of usage data was implemented and tested in parallel to the User Testing, see Section 5.

To asses the validity of the approach a comparison between results from data collection of data from tasks and result from the web survey was done, see Section 6. By doing this it was possible to evaluate how well an automated data collection method compares with a web survey, which is conducted to understand user interaction. This is final step in our method. Due to space restrictions, we cannot present the full results and refer the interested reader to [16].

4 Identification of Knowledge Needs

To identify the company employees' point of view on their need of customer understanding a workshop was conducted. Invited to the workshop were parts of the support team, test team, development team, and the product owner. This set up of participants was selected to get a wide range of ideas and to enhance discussion, as people from different teams are very likely to have different standpoints.

The goal of the workshop was to gain more insight of participants perception of the system and their ideas on knowledge needs about user interaction with the system that could be used as a foundation for defining the questions that should be answered by the automated data collection.

The workshop had seven participants from the company. It started with a brief introduction to the idea of this study and the goal and structure of the workshop, this lasted for about ten minutes.

The participant were then presented with questions, one at a time, which all were first described for about two minutes so that everyone understood the question. After a question was introduced each participant wrote down his or her ideas to the question on sticky-notes, which were collected and put on a whiteboard. The time limit, which participants had while writing down their ideas, was strictly five minutes. The reason of not letting them work in groups, at this point, was to avoid them from influencing each other and decreasing productivity.

The notes that were similar to each other were then grouped, and simultaneously there was an open discussion about each group of notes. The reason for this structure during the workshop was to first let people think alone and then to have an open discussion to reason about their answers and see if more could be extracted, each open discussion had a time limit of ten minutes. By grouping the questions it was easier to have a structured discussion and see which points seemed to be common perceptions between participants.

In total, for all questions, 74 sticky-notes were written. The complete workflow of the workshop session can be seen in Figure 2. The questions for the workshop were designed in such a way that they together would give a deeper knowledge of what was known, what was not known, and what was interesting to know. This knowledge could then be used to define a number of questions that should be answered by the survey and the data collection.

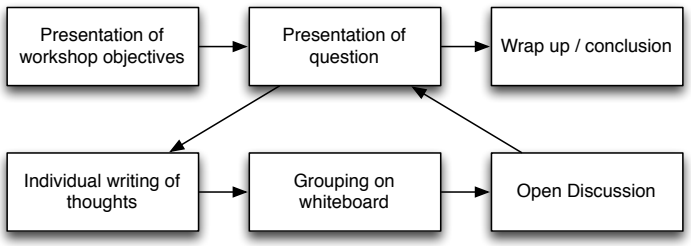

Fig. 2. Workflow of the workshop process

In the following, we review two questions of the workshop and the results. The third question dealt with porting the software to a different platform which is out of scope for this paper.

Which parts of the software need improved user experience? The purpose of this question was to understand what the employees at ATEA Global Service saw as points of improvement in regards to the user experience. By questioning them it would be possible to predict the outcome of the gathered data and understand if they are able to predict the need of their end user. This question was the foundation for defining which parts of the software were important to monitor and analyze. The sticky-notes were gathered and grouped into five different categories as shown in Table 1. The diversity of participants was very important for the discussion of each category and it resulted in a clear view of possible parts to monitor and analyze.

What would you like to know about the users' interaction with Accelerator? By asking this question, it was possible to find points where the employees of ATEA Global Services felt uncertain in regards to how the software is used. The idea was also to understand if parts of the software had been developed and maintained without clear and motivated reasons. The question was defined as guiding purpose to which data that should be gathered and extracted for analysis. The categories shown in Table 2 have been identified.

From the open discussion, it was clear that participants where uncertain of how the product was being used. Which seems to come from a lack of collaboration with the actual end users at the customer companies. It was also clear that the participants found this question interesting and wanted to know more

What happens?	The transparency of the software towards the end-user. Example of sticky-notes: "Orders: What happens with the order when you complete your order."
Grouping	The possibility to group and simplify some procedures. Example of sticky-notes: "The need to go to different parts of the system to order different types of products."
Admin	The parts that are related to the Admin interface and advanced configuration.
Menu	The ideas discussing the navigation. Example of sticky-notes: "Menu system, there is a limited space"
Customization	The reasoning about a more personalized interface. Example of sticky-notes: "An Accelerator for different users' roles."

Table 1. Question categories on required improved user experience.

Misunderstandings	See where end user have issues to use the product. Example of sticky-notes: "What makes a user confused, regards to how the software works"
Statistics	Information based on descriptive statistics. Example of sticky-notes: "Least used part of the software"
Time	Information based on time, when and how end user use the software. Example of sticky-notes: "How long time does it take a user to complete different tasks"
Frequency	How often and how is the software used. Example of sticky-notes: "How big is the user group of daily users"

Table 2. Question categories on knowledge required about user interaction.

about their end users, as there were so many different ideas. The result from this questions was in the following used to define what data to gather.

4.1 Identified Data Analysis Needs

There were some obvious outcomes from the workshop, first of all there is a need for increased communication within the organization. It was also clear that many felt uncertain about their end users usage of the software. This emphasizes the need of more statistical data of the software use or increased communication with end users.

By gathering and analyzing the information from all Workshop Questions the following eleven questions were extracted and defined to possibly be answered by the survey and the automatic data collection:

1) What is the frequency of use for functionality: By answering this question usage and possibly user interface design decisions can be made better. It defines which parts should be prioritized when evolving the product.

2) Frequency of use for different roles: The reason for understanding frequency of use for different role is so that the user interface could be tailored for some roles which use the system a lot.

3) Are there any functions that are not used: The question aims at reducing complexity by removing not used functionality.

4) How is the internal search of the website utilized: Since many parts of the software are based on search it is interesting to know if the search functionality is used once within a session or several times. If search mostly occurs once it might be possible to suggest a simplified design. Furthermore, the system supports a test based search and a tree based search.

5) How long time does it take a user to complete a task: If a part of the software is complex, it most probably will take longer time to complete task. By understanding complex parts, it would be possible to pinpoint where to focus on improvement or parts that should be revised.

6) How does software use differ between regular and non-regular users: Based on this question, strategic decision can be made from the data. For example, which type of user should be prioritized.

7) Which task is the most difficult to complete: By understanding what scenario and part of navigation that are difficult for the end-user, it is possible to see where it is most vital to simplify the user interface.

8) Are there any trends, between different versions: Changes over time are important feedback for continues development and enables an evaluation of previous decisions as suggested in the BLM loop.

9) Are there parts of the software with a high bounce rate: By finding parts of the software with a high bounce rate it is possible to revise them. A bounce is when a user navigates to a certain part and then navigates to another one in just a matter of second. This indicates that the former was not what the user was looking for. A high bounce rate can be a sign of design or content issues.

10) How often and where do drop offs occur: A drop off occurs when a user works on a task and then aborts. By finding drop-offs, it could be possible to pinpoint parts that have a complexity that is too high or where there are too many steps and options for the end-user to complete.

11) What time of the day are tasks carried out: By understanding when the task is started and what is carried out, it might be possible to motivate what parts should be extracted to a handheld device as these may be used more often in the mornings, evenings or during lunch.

4.2 Question refinement

In the following, we describe how the questions defined in Section 4.1 are redefined in a more detailed manner and broken down into a tree structure. Moreover, the questions are analyzed to identify which could be answered by automated data collection and respectively a web survey.

Each question has been broken down further and annotated by notes which indicate whether an answer to a question is feasible by using the automated

data collection or by using a survey. Typically, the workshop questions deal with user impressions and are therefore not feasible to answer by an automated data collection approach but with an online survey whereas the broken down questions are on a level which can be answered by automated data collection but maybe not with a survey. We use «Survey» for questions answerable by a survey and those answerable by automated data collection with «DataCollection». We focus in the following on the workshop question *7) Which task is the most difficult to complete* and refer to [16] for the full description of all breakdown questions.

Breakdown Question 7 is the same as question 7 from the workshop as presented in section 4.1. For other workshop questions this might be different. The purpose of this structure is that a survey could answer Breakdown Question 7 directly by asking the participants how difficult each task is. However, to design a survey question for the two refined questions, 7.1 and 7.2, would be to complex. These two questions try to identify which task are difficult to complete by analyzing the longest time it takes a user to complete a step during the completion of a workflow, represented as question 7.1, and by looking at the task which has the highest value of completion time per steps, represented in question 7.2. Both can be answered automatically from the collected data.

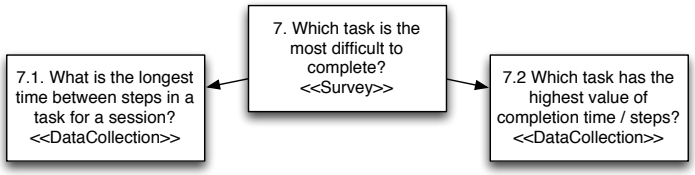

Fig. 3. Breakdown Question 7

As an outcome of this activity the initial 11 questions have been refined into 36 refined questions and for each question, it was defined whether they can be answered by an automated data collection or a web survey.

5 Automated Data Collection

The user interaction is collected as a set of traces, where a trace is a record of a single user interaction with the system.

The process of developing the data collection included three steps: First, determining what data needs to be collected. Second, selecting an appropriate technology for implementing usage tracing. Finally, executing the implementation of usage tracing with the selected technology to capture the user interaction and to answer the questions presented in Section 4.2. Each stored user interaction trace includes which part of the system is used, who is the user, what is the user's action (e.g., a button click), the roles of the user, and time related information like timestamp and execution time.

As the user interactions are handled by a wide variety of source files, the addition of data collection would require many manual changes. However, as tracing is a true cross cutting feature and the source code closely followed guidelines about naming of methods, aspect oriented programming [17] was used to introduce the usage tracing code into the system. We used 11 point cuts to cover the complete usage tracing by introducing the code into all methods which dealt with user interactions in the system like all onClick-methods, which are called every time a user clicks on an user interface element. We were able to exploit company specific source code naming conventions.

The usage of aspect oriented programming enabled a fast development of the data collection code. Furthermore, it supports the automatic evolution of the usage tracing in case new functionality is added which follows the same guidelines as the existing code.

6 Preliminary Evaluation

This section describes a preliminary evaluation that the solution in section 5 produces knowledge that conforms to the users perception expressed in a web survey. The evaluation was performed by letting 15 test subjects take part in a user testing workshop. In the workshop, each user was asked to perform the set of tasks shown in Table 3.

1. Request Hardware	User ought to complete a request for a computer.
2. Change Password	User ought to successfully change the password.
3. Request Software	User ought to complete a request for a software product.
4. Request Access	User should request membership to a user group.
5. Approve Order	User should approve a request, while using a manager account.
6. Cancel Order	User should cancel a request.

Table 3. Tasks executed by the test subjects in a random order.

These six tasks cover the essentials of the case product and what users most often use the product for according to ATEA Global Services. Since requests of different kinds are the idea of the case product three tasks were used to cover this. Requests need approval in some cases and due to this the Approve Order and Cancel Order were selected. However, due to issues with the automated data collection, these two tasks could not be measured and thus are excluded from the following description. To also cover some of the manage user part of the product one task was designed for changing password.

As mentioned in Section 3 the tasks were designed to reflect a real world scenario. With this in mind the task were formulated in such way that they represent a possible scenario that end-users could encounter. To counter undesired variations related to learning- and boredom-effect the ordering of tasks for each test subject was made so that no subjects performed the tasks in the same order.

After completion of the tasks, the users were sent an e-mail with a link to a web survey. After the data collection had been run on the user testing data

and the data from the web survey had been collected, the results were compared. Four Breakdown Questions were selected, number 1 (Time for a user to complete a task), 3 (Product bounce rate), 5 (Utilization of internal search) and 7 (Most difficult task to complete). We will report about the last one and refer to [16] for the others. The web survey question related to this Breakdown Question was: "Rate how difficult each task was to complete on a scale from 1 (easy) to 4 (very difficult)."

The most difficult task was Change Password which was rated 2 on average. Then came Request Access with 1.72 followed by Order Hardware and Order Software with 1.52 and 1.3.

The corresponding measurements are the average completion time per step for each task. The highest average completion time per step by far was 47.4 seconds for the Change Password task. Then came Order Hardware (14.9 seconds) and Order Software (9.0 seconds). Finally, Request Access had an average completion time per step of 8.3 seconds. These three tasks have a similar small average completion time per step in comparison with Manage Account.

When we compare the average completion time per step with the survey results, we see both the measurement as well as the survey indicate that Change Password is the most difficult task. This was also seen during the experiment as the test subjects struggled choosing a password which conforms to the password rules which were enforced but not explained by the system. Request Access was rated higher in the survey than indicated in the measurements. The ranking between Order Hardware and Order Software was the same in measurements and the survey. Kendall's tau is 0.333 indicating a weak positive correlation. We refer to [16] for more details about the other parts of the evaluation including results of statistical tests.

The measurements for the three other tasks than Change Password are very similar. Thus, small changes in the measurements may lead to different ranks and also the users might not be able to judge the small differences. So, we plan to do follow-up evaluations with more subjects.

7 Related Work

Automated user interaction analysis is well known in the web development with Google Analytics as one of the major platforms. As an example, Hasan et al. report about using Google Analytics for e-commerce systems [18]. Google analytics and similar systems, however, have the problems that the data collection and analysis is done on systems not owned by the company which raises privacy issues. Furthermore, the data collection is constrained by the supported features.

Van der Shuur and Jansen [19] have presented a solution for improving software quality by automatically gathering and reporting how a software service is being used. The data gathering was implemented in the service layer using aspect-oriented programing as in our approach. Furthermore, the reporting was built using a set of metrics that were concerned with quality attributes like availability, accuracy, reliability and usability. They concluded that their solution was

expected to contribute to an increase in software quality and that future work was needed on how to use data mining techniques for reporting on software utilization.

For usage tracing aspect-oriented programming stands out as having been tested for its suitability when implementing automated data collection for usability evaluation and usage tracing. Tart and Moldovan showed that it could be used for automated usability evaluation [20] and equal results where gained by Tao who used the same framework [21]. Tarby et al. compared aspect-oriented programming with Agent-Based Software Architecture concluding that they could be used as complement. The recommendation was to use aspect-oriented programming for defining traces while the agents would be used to "produce traces whose visualization will be made in real time" [22]. A trace is referred to as a record of an action performed by a user.

However, these papers lack a comparison against other data collection techniques compared to this paper.

8 Conclusion and Future work

The presented work is concerned with enabling software companies to measure and subsequently learn how their customers use their software as a precondition for improving the software in such a way that it benefits the customer. We used a web-based portal software developed by ATEA Global Services as an industrial case.

The different aspects, which the company were interested to know about how user interact with the software, were identified in a workshop with different roles. An automated data collection system for the identified aspects was built and in a follow-up experiment the results of the automated data collection system was compared to answers of the test subject in a survey. We refer to [16] for the complete results and further details of our research study.

Future works include a follow-up experiment with a higher number of test subjects as well as monitoring the system and its measurements over several versions to identify whether the added automated data collection capabilities enable the company to continually improve the software.

References

1. Olsson, H.H., Bosch, J.: Towards data-driven product development: A multiple case study on post-deployment data usage in software-intensive embedded systems. In: LESS. Volume 167 of Lecture Notes in Business Information Processing., Springer (2013) 152–164
2. Olsson, H.H., Bosch, J.: Post-deployment data collection in software-intensive embedded products. In Herzwurm, G., Margaria, T., eds.: ICSOB. Volume 150 of Lecture Notes in Business Information Processing., Springer (2013) 79–89
3. Ries, E.: The Lean Startup: How Constant Innovation Creates Radically Successful Businesses. Penguin Group, London (2011)

4. Highsmith, J., Cockburn, A.: Agile software development: The business of innovation. IEEE Computer **34**(9) (2001) 120–122
5. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New directions on agile methods: A comparative analysis. In Clarke, L.A., Dillon, L., Tichy, W.F., eds.: ICSE, IEEE Computer Society (2003) 244–254
6. Morris, M., Dillon, A.: How user perceptions influence software use. Software, IEEE **14**(4) (Jul 1997) 58–65
7. Soloway, E., Guzdial, M., Hay, K.E.: Learner-centered design: the challenge for hci in the 21st century. Interactions **1**(2) (1994) 36–48
8. Argyris, C., Schön, D.: Organisational learning: A theory of action perspective. Addison Wesley (1978)
9. Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M.: Controlled experiments on the web: survey and practice guide. Data mining and knowledge discovery **18**(1) (2009) 140–181
10. Bosch, J.: Building products as innovation experiment systems. In Cusumano, M.A., Iyer, B., Venkatraman, N., eds.: ICSOB. Volume 114 of Lecture Notes in Business Information Processing., Springer (2012) 27–39
11. Dzamashvili-Fogelström, N., Gorschek, T., Svahnberg, M., Olsson, P.: The impact of agile principles on market-driven software product development. Journal of Software Maintenance **22**(1) (2010) 53–80
12. Sommerville, I.: Software Engineering, 6th edition. Pearson Education: Essex, England (2001)
13. Bird, C., Murphy, B., Nagappan, N., Zimmermann, T.: Empirical software engineering at microsoft research. In Hinds, P.J., Tang, J.C., Wang, J., Bardram, J.E., Ducheneaut, N., eds.: CSCW, ACM (2011) 143–150
14. Beck, K.: Test Driven Development: By Example. Addison-Wesley Professional (2002)
15. Cockburn, A.: Agile Software Development. Addison-Wesley Professional (2001)
16. Backlund, E., Bolle, M.: Automated usage tracing and analysis: a comparison with web survey. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden (2013)
17. Kiczales, G., Lamping, J., Mendhekar, A.: Aspect-oriented programming. ECOOP'97 - Object-Oriented Programming **1241** (1997) 220–242
18. Hasan, L., Morris, A., Probets, S.G.: Using google analytics to evaluate the usability of e-commerce sites. In Kurosu, M., ed.: HCI (10). Volume 5619 of Lecture Notes in Computer Science., Springer (2009) 697–706
19. van der Schuur, H., Jansen, S., Brinkkemper, S.: Becoming responsive to service usage and performance changes by applying service feedback metrics to software maintenance. In: Proc. of the 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops, Ieee (September 2008) 53–62
20. Tarta, A., Moldovan, G.: Automatic usability evaluation using aop. In: Automation, Quality and Testing, Robotics, 2006 IEEE International Conference on. Volume 2., IEEE (2006) 84–89
21. Tao, Y.: Capturing user interface events with aspects. In: Human-Computer Interaction. HCI Applications and Services. Springer (2007) 1170–1179
22. Tarby, J., Ezzedine, H., Kolski, C.: Trace-Based Usability Evaluation Using Aspect-Oriented Programming and Agent-Based Software Architecture. Human-Centered Software Engineering (2009) 257–276