Conception and Implementation of a Usability Problem Analysis Tool and its Comparison with another Usability Technique on the Example of a Workflow-driven Integration Environment

MASTER'S THESIS

Bonn, den 17. Juli 2013

in cooperation with:

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III
Professor Dr. Armin B. Cremers

OLIVER SEEBACH | UNTERSTRASSE 19 | 56814 FAID | OLIVER.SEEBACH@DLR.DE

Master's Thesis
"Conception and Implementation of a Usability Problem Analysis Tool and its Comparison with another Usability Technique on the Example of a Workflow-driven Integration Environment"
By Oliver Seebach
Rheinische Friedrich-Wilhelms-Universität Bonn
Submission: 17.07.2013

First Supervisor
Prof. Dr. Armin B. Cremers
Rheinische Friedrich-Wilhelms-Universität Bonn
Institute of Computer Science III

Second Supervisor
Prof. Dr. Claudia Müller-Birn
Freie Universität Berlin
Institute of Computer Science

In Cooperation with
German Aerospace Center (DLR)
Simulation and Software Technology
Distributed Systems and Component Software

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind.

Bonn, den 17.07.2013

Abstract

The demand for good usability has become indisputable as usable products are more profitable, secure and enhance the reputation of its producing company.

For many applications abstract usability problems like "Users have problems using the online shop" are known, but bringing in concrete design improvements is hard. Thus, it would be desirable to transfer it to concrete problems like "The meaning of text field 'ID' in form 'Payment' is unclear". Deriving concrete redesign ideas from such a concrete problem is not that hard anymore.

The thesis provides a concept and a prototypical implementation of a tool to analyze an existing abstract usability problem to enable an analyst to submit substantiated improvement suggestions. The tool is automatically and remotely applicable. This reduces travel expenses and accelerates data collection. To evaluate the results the proven Think Aloud method is taken as reference usability technique. The Track-Analysis-Tool is applied on the example of the workflow-driven integration environment RCE. A user task representing the abstract usability problem "Users have problems getting started with RCE" is defined. A study with 13 participants divided into two iterations is conducted.

The investigation of the results shows that the Track-Analysis-Tool reveals the same crucial usability problems as the Think Aloud method does and is particularly useful when it comes to quantifications like task completion times. The Track-Analysis-Tool lacks to gather subtle usability problems which can be detected with other techniques like the Think Aloud method.

Überblick

Der Bedarf nach Benutzerfreundlichkeit ist nicht von der Hand zu weisen, da benutzerfreundliche Produkte ertragreicher und sicherer sind und das Ansehen des Herstellers erhöhen.

In vielen Anwendungen sind abstrakte Probleme wie "Benutzer haben Probleme mit dem Onlineshop" bekannt, jedoch ist es schwierig konkrete Designverbesserungen einzubringen. Daher ist es wünschenswert, daraus konkrete Probleme abzuleiten, z.B. "Die Bedeutung des Textfeldes 'ID' im Formular 'Zahlungsweise' ist unklar". Von solch einem konkreten Problem lassen sich Designänderungen leicht ableiten.

In dieser Arbeit wird ein Konzept und eine prototypische Implementierung eines Tools zur Analyse von existierenden abstrakten Problemen vorgestellt. Somit kann ein Analyst fundierte Verbesserungsvorschläge einbringen. Das Tool ist automatisiert und per Fernzugriff einsetzbar. Dadurch werden Reisekosten gespart und die Datenerfassung beschleunigt. Zur Evaluation wird die bewährte Think Aloud Methode als Referenztechnik zu Rate gezogen. Am Beispiel der Integrationsumgebung RCE wird das Track-Analyse-Tool erprobt. Dazu wird eine Aufgabenstellung definiert, die das abstrakte Problem "Benutzer haben Probleme mit den ersten Schritten in RCE" abdeckt. Eine Studie mit 13 Teilnehmern - aufgeteilt in zwei Iterationen - wird durchgeführt.

Die Untersuchung der Ergebnisse zeigt, dass das Track-Analyse-Tool die gleichen kritischen Probleme wie die Think Aloud Methode erkennt und sich als besonderes sinnvoll erweist, wenn Quantifizierungen erwünscht sind, z.B. in Form von Bearbeitungszeiten. Allerdings scheitert das Track-Analyse-Tool darin, subtile Probleme zu entdecken. Diese können mit anderen Techniken wie der Think Aloud Methode erkannt werden.

Acknowledgements

First of all I would like to express my gratitude to Prof. Dr. Armin B. Cremers from Rheinische Friedrich-Wilhelms-Universität Bonn and Prof. Dr. Claudia Müller-Birn from Freie Universität Berlin for supervising my thesis.

I would also like to thank my tutors who helped me to keep on track and provided excellent guidance. My appreciations go to Doreen Seider and Robert Mischke from Simulation and Software Technology of the German Aerospace Center, to Mark von Zeschau from Rheinische Friedrich-Wilhelms-Universität Bonn and to Julia Schenk from Freie Universität Berlin. Moreover I would like to thank the entire RCE development team.

My sincere thanks also goes to the probands of the study I conducted in the context of the thesis. Thanks to the employees of Simulation and Software Technology and the Institute of Air Transport and Airport Research of the German Aerospace Center for spending their valuable time.

Last but not least I would like to thank my beloved ones for supporting me throughout the entire time.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

AOP Aspect Oriented Programming

AWT Abstract Window Toolkit

CPACS Common Parametric Aircraft Configuration Schema

DLR German Aerospace Center (German: Deutsches Zentrum für Luft- und Raumfahrt e.V.)

EPL Eclipse Public License

GC Graphical Context

GEF Graphical Editing Framework

GOMS Goals, Operators, Methods, Selection rules

GUI Graphical User Interface

HTML Hypertext Markup Language

IDE Integrated Development Environment

IPO Input-Process-Output

ISO International Standards Organization

JSON JavaScript Object Notation

JVM Java Virtual Machine

OSGi Open Services Gateway Initiative

PDF Portable Document Format

RCE Remote Component Environment

RCP Eclipse Rich Client Platform

SUM Single Usability Metric

SWT Standard Widget Toolkit

Chapter 1

Introduction

"It is a common experience that a problem difficult at night is resolved in the morning after the committee of sleep has worked on it."

—John Steinbeck

This introduction helps the reader to understanding the motivation for the thesis and introduces the setting, challenges and limitations. To lead over to the next chapters, the structure of the thesis is outlined.

1.1 Motivation

The demand for usability testing has become indisputable. Software that meets its users' needs makes it more efficient, effective and pleasant. That means more tasks are solved correctly, tasks are solved faster and users are more satisfied which enhances the reputation of a product. Moreover, improving the usability of software can save money: If a website is more usable it is likely that it will make more sales. If a machine is more usable it is likely to be more secure. If a highly paid engineer saves a few minutes to accomplish each task, the investment into usability testing will pay off quickly [BM05, p. 17].

There are many techniques proposed to enhance the usability of a product. They can be divided into evaluator methods where a usability expert investigates the product in a certain procedure and test methods where actual or prototypical users are involved in the procedure. Many usability problems can already be detected with a minimal number of probands [Nie94a]. It can often be useful to apply several techniques to find as many usability problems as possible and to look at them from several points of view.

Analysis is often based on logged data [Nie92b], but there has no approach been proposed yet that works robust in the context of rich client

applications. As a large amount of data is collected it is often not possible to be interpreted without additional contextual knowledge. The raw data needs to be filtered and aggregated with respect to the context. The desired result is a reasonable amount of data a human being is able to read, analyze and interpret. Thus, deeper insight into problems can be derived and redesign suggestions can be proposed.

In the thesis it is explored which difficulties appear when integrating such an approach into a rich client application. Furthermore, it is analyzed to which extent such an approach can replace or supplement other proven usability techniques on the example of the *Think Aloud* method.

The approach is applicable remotely and in an automated fashion. This enables the approach to include many users in a distributed environment. Different departments of a company throughout the world can participate in the usability test without causing enormous travel expenses.

The thesis focuses on the analysis of usability problems when working for the first time with the Remote Component Environment (RCE) [DLR13c] developed at the *German Aerospace Center* (German: Deutsches Zentrum für Luft- und Raumfahrt e.V., DLR). From people who used RCE for the first time it is known that a better start into RCE is desired. The analysis of usability problems will be applied to this abstract problem.

Deeper insight about the abstract usability problem is developed as it is analyzed on a finer granularity - this will help to understand the cause and to propose redesign solutions. Additionally the results can substantiate the reasoning for the realization of a specific solution.

The application on the first steps with RCE will provide a better access into RCE for engineers. Since RCE is open source software, this benefit is not limited to the borders of the DLR. The DLR is an important part of the setting which is presented next.

1.2 Setting

The Master's thesis is written at the German Aerospace Center in the facility *Distributed Systems and Component Software*. The German Aerospace Center is the national aeronautics and space research center of the Federal Republic of Germany. Research is done in the fields aeronautics, space, energy, transport and security. DLR has approximately 7400 employees at 16 locations in Germany: Cologne, Augsburg, Berlin, Bonn, Braunschweig, Bremen, Goettingen, Hamburg, Juelich, Lampoldshausen, Neustrelitz, Oberpfaffenhofen, Stade, Stuttgart, Trauen, and Weilheim. DLR also has offices in Brussels, Paris, Tokyo and Washington D.C. [DLR13b]

The facility Distributed Systems and Component Software describes its work as follows:

> "The Distributed Systems and Component Software department works on software engineering, develops challenging software solutions for the German Aerospace Center, and does research in scientific projects in the field of software technology. The main focuses of work are in software engineering, high performance computing, and distributed software systems." [DLR13a]

RCE is based on the *Eclipse Rich Client Platform* (RCP). It is an distributed platform for the integration of applications. To these applications RCE provides access to software components like a workflow engine, privilege management or an interface to external resources. Thus, RCE enables application developers to concentrate on the application-specific logic rather than on the interaction with other applications by embedding them into one unified environment. [DLR13d]

RCE is open source software and can be accessed and used for free. It is already in use among several departments in DLR and also external companies have shown interest in using RCE.

The probands to conduct the usability tests are employees of the DLR. There are no preferences regarding age, gender or expertise in terms of informatics. Since the process of getting started with RCE is under observation, the probands are required to have no prior knowledge about how to work with RCE.

After the setting and the motivation is defined, the problem definition is presented.

1.3 Problem Definition

The problem under observation is to develop an approach that is capable of analyzing a known usability problem in more depth in an automated fashion. Having the result of the analysis at hand an analyst should be able to quickly detect detailed usability problems and derive redesign solutions for them.

The data basis for the approach is formed by captured interaction of the user with the interface.

To evaluate the approach another proven usability technique is taken as reference and simultaneously applied in a user study. Differences and commonalities of results of the two methodologies are compared regarding the identified detailed usability problems. Thereby the capabilities and limitations of the usability Track-Analysis-Tool based on captured interaction are explored.

Design improvements are derived from the identified usability problems and realized. Suitable metrics are determined. They are measured within the user study before and after the design improvements are realized. To show that actual usability problems are detected and solved it is desirable that the metrics improve.

Having this problem definition in mind, research questions are formulated.

1.4 Research Question

Research questions derived from the problem definition are the following:

- If an abstract usability problem is identified, is there a way to deeply analyze this problem in an automated fashion?
- Is it possible to gain information about usability problems using captured interaction data, such as mouse tracks or clicks?
- If it is possible, does it yield the same results as other proven usability methods can extract?
- If the methods yield different usability problems, which ones are more crucial?
- Is there a systematic reason why they do or do not find the same usability problems?
- Can the usability of the given software be significantly improved by enhancing the problems found by the newly developed approach?

The answers to these research questions imply several challenges.

1.5 Challenges

There are several challenges to face. Some of them are of conceptional nature while others are technical challenges.

An approach has to be developed that allows to associate user actions with intentions given additional context knowledge. The procession of potentially overlapping intentions in a robust manner is a particularly challenging task.

The rating of each section of the overall task has to be defined. Therefore existing metrics and newly developed metrics have to be weighted and reviewed for suitability.

Moreover, it has to be made sure, that enough events can be logged so that substantiated statements can be made. Therefore it has to be clarified how comfortable the interaction with the used technologies is.

The logging functionality should be integrated into the software in a non-intrusive manner. That means it should always run in background without interfering with the user's actions and it should be activated knowingly. Moreover it should be plugged into the application without interfering with the source code. Similar to other cross-cutting concerns like logging or security is should not be spread over multiple classes.

From an organizational point of view it is a challenge to acquire enough suited probands for the user study. As learning effects must not occur, each user can only participate in one iteration. Therefore an appropriate task, covering the required topics has to be defined.

It is assumed that the correct assignment of interactions to intentions in a robust and automated manner is a non-trivial task.

As a further challenge, another usability technique suitable to compare identified usability problems has to be chosen. Due to the temporal constraints of the thesis it is not possible to test all methods. Thus, research based on the literature has to suffice and substantiate the decision.

As the author has just limited experience in conducting studies, the preparation, execution and analysis of the study is associated with certain risks and uncertainties.

If the detected problems of both techniques turn out to be diverse, an approach to deal with that has to be developed. Moreover, the derivation of solutions to the identified problems is a challenge. The problems are not predictable, thus the complexity of the solution generation is not known in advance.

1.6 Limitations of the Thesis

For reasons of time and scope there are several limitations to the thesis:

To recall, the presumed general existence of a problem must be clarified. Though there are methods proposed [VSK96][Nie92a] to generically identify usability problems, an integration of them would exceed the scope of the thesis. An automated, generic approach that identifies problems without prior knowledge is not the goal of the thesis.

Another limitation of the thesis is the number of iterations of the usability test process. To allow for comparison of metrics, at least two iterations must be executed, but temporal limitations might prohibit further iterations.

Moreover the granularity of examination might cause some trouble. Though some iterations of this approach might optimize a wizard, it would possibly not catch that the problem is the wizard itself. Thus, there is no guarantee that the approach will not run into a local max-

imum (an improved wizard) instead of a global maximum, replacing the wizard by another more suitable concept.

The reader also has to keep in mind that the user study just consists of one task, i.e. the successful execution of a workflow. In order to examine the limitations and possibilities of this approach more exhaustive studies have to be conducted.

A limitation that rises from the setting of the thesis refers to the solution generation. The proposed solutions must not disturb functionalities of the software other than the successful execution of the first workflow.

Furthermore the reader has to keep in mind that the number of subjects is rather small. Thus, empirical statements cannot be made. But as Nielsen [Nie94a] states, even small numbers of subjects are sufficient to identify the majority of usability problems. As the scope of the thesis is the detection and not the quantification of usability problems, a small number of users is assumed to be sufficient.

1.7 Outline

The thesis is structured as follows:

- 2—"Foundations": Foundations, which are vital to understanding several technical and conceptional parts of thesis are explained in this chapter.
- 3—"Conception": In the chapter "Conception" the developed algorithm, the criteria for the choice of a reference usability technique and metrics are described.
- 4—"Implementation of the Track-Analysis-Tool": This chapter describes the technical infrastructure and the use of the exchange data format *JSON*. Moreover the implementation of the tool generating logs from RCE and the analysis tool, implementing the developed algorithm are presented.
- 5—"Usability Study": In the chapter "Usability Study" the setup, the defined user task and the questionnaires to measure satisfaction metrics are described. More over reasons for the chosen reference method, the *Think Aloud* method are stated. The iterative approach and the realized design improvements are explained.
- 6—"Discussion of the Results": In the discussion the identified problems of the two methods are compared. The measured metrics are discussed. Commonalities and differences from the Track-Analysis-Tool and the Think Aloud method are presented.
- 7—"Related Work": In the chapter "Related Work" scientific efforts dealing with the same context are summarized and analyzed with regard to the scope the thesis.
- 8—"Conclusion and Future Work": This chapter summarizes the thesis and gives an outlook about possible future work.

Chapter 2

Foundations

"The foundation stones for a balanced success are honesty, character, integrity, faith, love and loyalty."

—Zig Ziglar

This chapter describes technical foundations and methodologies that are vital to understand several aspects of the thesis. This contains the notion of usability as it is understood in the context of the thesis. An introduction to the *Think Aloud* method which is applied in the study is given. To understand the technical background, the application under observation (RCE) and the technologies it is based on (RCP, JFace and SWT) are presented. It is always a creative process to derive redesign suggestions to a given problem. In the course of the thesis the creativity method of the *Six Thinking Hats* is applied and introduced here.

2.1 Usability

This section aims to give an overview about usability in general and a notion how it is understood in the context of the thesis.

2.1.1 Definitions of Usability

There are several definitions of usability proposed in literature. Hereinafter some of them are cited and the crucial properties, particularly in the context of the thesis, will be extracted to form the understanding of usability vital for the thesis. Shackel and Richardson [SR91, p. 24] define usability as

> "The capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and user support to fulfill the specified range of tasks within the specified range of environmental scenarios."

Nielsen states his understanding of usability in [Nie94b, p. 26]:

> "Usability has multiple components and is traditionally associated with these 5 attributes: [...] Learnability, efficiency, memorability, errors, satisfaction."

According to Rubin [RCS11, p. 4] a product is usable when is has a good usability. For him, this is given when

> "... the user can do what he or she wants to do the way he or she expects to be able to do it without hindrance, hesitation or questions."

With ISO 9241-11 the International Standards Organization (ISO) has published a definition of usability as being

> "... the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

In the thesis the author focuses on usability as the combination of the following aspects:

- Effectiveness: Did the user succeed with the task? Did the user need hints to succeed?
- Efficiency: How long did it take the user to succeed with the task?
- Satisfaction: How satisfied was the user? Were there any frustrating experiences?
- Constraints: Possible constraints are a certain target user group, a certain limit of required training, social, cultural, political or monetary constraints.

Derived from that, enhancing usability means to maximize efficiency, effectiveness and satisfaction without violating these constraints.

2.1.2 Relation and Distinction to User Experience

The terms *usability* and *user experience* are often used imprecise and differently [Pro10][Not01][TA10, p. 4][LRV+08]. Thus, the relation and a distinction between the terms as used in the context of the thesis is clarified here.

Law et al. collected a sample of definitions of the term *user experience* [TA10, p. 4][LRV+08]. The definitions mentioned above show that the understanding of the terms vary a lot.

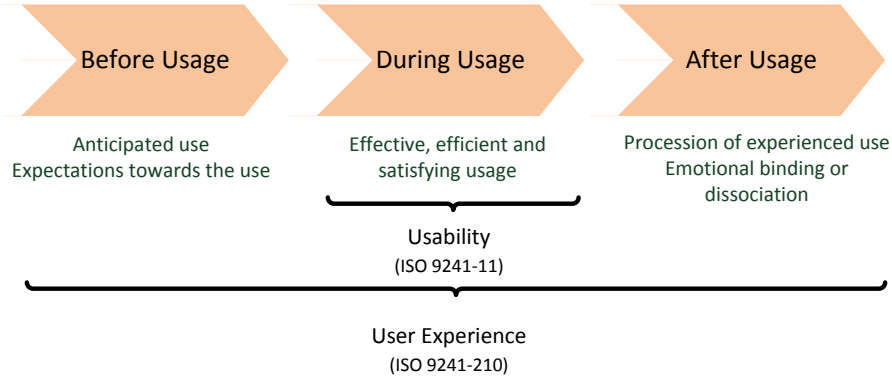

Figure 2.1: Relation between usability and user experience [Pro10]

Figure 2.1 depicts how the relation between usability and user experience can be seen.

Derived from [TA10, p. 4] the following distinction is made: While usability refers to the ability of the user to use the product for the task successfully, user experience covers a broader view. User experience covers the entire interaction with the thing including thoughts, feelings, reputation and perceptions brought about by the interaction with the product. A good example for the distinction is the unpacking of the product.

While usability ignores it and focuses on the actual use of the product, user experience also examines the act of unpacking the product as it already influences the overall attitude towards the product.

2.1.3 Further Considerations

In this section further considerations when talking about usability are described.

Target Groups

When considering usability it is always a good idea to keep the target group in mind as it has a major impact on the results. [Nie89] as reported by [Nie94b] showed in a study that the difference between the individual users sad a major impact on the results, even more than the difference between the tasks.

There is a methodology proposed to classify users in a structured way [PHR+86]. Moreover, it was shown that good designs often turn out to be suitable for many user groups.

Nielsen proposes to determine the target groups by the experience [Nie94b, p. 43 f]. Therefore three dimensions of experience are considered:

- Experience regarding the current interface of the product
- Experience with computers in general
- Experience with the domain the product deals with

Further criteria for classifications are proposed by Cooper et al [CRC12]:

Based on demographic criteria like age, gender, educational level or home zip code users can be classified [CRC12, p. 69]. Goal-directed design is elaborated as another approach to classify users based on their goals [CRC12, p. 3]. These goals can be divided into experience goals like feeling in control or having fun, end goals like being able to create a workflow unaided and life goals like living a good life [CRC12, p. 92]. Moreover, personas can also be used to model user groups [CRC12, p. 82].

In the context of the thesis the experience regarding RCE is the only criterion to determine whether a user is a potential proband for the study or not. The probands are required to have no previous knowledge regarding the operation of RCE.

Trade offs

Referring to Nielsen [Nie94b, p. 41 f] it is often necessary to make trade offs when creating redesign ideas, as the following arguments show:

Expert users often need more functionality in software than novice users. This fact and the idea to combine the functionality needed by both groups into one interface can lead to complex interface. These complex interfaces can be a problem themselves. It can be a complicated task not to confront novice users with the expert functionality and vice versa.

It is often a good idea to enable several ways to interaction, e.g. via double-click and additionally a menu entry on the respective element. Having this in mind, interfaces can often satisfy both novice and expert users. It is a good approach to first try to find a win-win situation where both user groups are satisfied. If that is not possible, the design should focus on the project specific requirements.

Metrics

In general, a metric is a way of measuring or valuating a particular thing or phenomenon [TA10, p. 7]. Metrics quantify a certain property or state of a thing in a numerical representation. So be able to compare metrics, there must be a proven procedure or methodology to measure this metric.

For various domains there is a specific set of metrics. For instance, in aviation metrics of an aircraft like drag, fuel consumption, passenger

capacity are of interest. In business revenues, costs and return on investment are typical metrics.

In usability there are also domain specific metrics. In contrasts to the metrics just mentioned, the metrics in usability describe how people interact with a product or system. As peoples' emotions and feelings are very different and often less tangible than in other domains, measuring these metrics has its own challenges. If it is intended to statistically evaluate and compare these metrics, confidence intervals are a good means. [TA10]

As a more exhaustive consideration would exceed the scope of the thesis, a short overview about existing metrics, derived from [TA10] is given.

The following list gives an overview about the different kinds of metrics and some examples:

- Performance Metrics: Task success, Time-on-task, Efficiency, Errors
- Issue Bases Metrics: Usability issues, i.e. events that prevent task completion, take the user "off-course", produce errors or create confusion
- Self-Reported Metrics: Post-task ratings, post-Session ratings measuring perception of the system or product
- Behavior/Physiological Metrics: Verbal and non-verbal expressions, eye-tracking, heart rate
- Combined/Comparative Metrics: SUM[1], usability scorecards

For more information, see [TA10] and [SDKP06].

After a general idea about usability is formed, the *Think Aloud* method is presented because it is the usability technique to be applied in the user study.

2.2 The Think Aloud Method

In this section the *Think Aloud* method is examined. Thereby some general information about the procedure is given, advantages and disadvantages are pointed out and a list of guidelines for the conduction of a study using the Think Aloud method is presented.

As an outlook, possible variations of the method are described.

[1] SUM stands for Single Usability Metric and combines several metrics, e.g. task success, errors and task time into one value

2.2.1 Origin and Approach

The Think Aloud method was first described by Lewis in 1982 [Lew82] as a method, having a test subject use the system while continuously think out loud. It was originally applied as a psychological research method [ES96]. There is the concurrent and the retrospective Think Aloud procedure [BJLP02]. In the following the focus is on the concurrent procedure.

The Think Aloud method is particularly applicable in the early phase of a product when it comes to the evaluation of the users' mental model about the product or system, but can also be applied in other phases. There are many cases when the Think Aloud method can be used to detect major usability problems with a very limited number of participants.

The following sections outline the advantages and disadvantages of the approach, based on the information from [RCS11, p. 204 f] and [Nie94b, p. 195 ff].

2.2.2 Advantages

The Think Aloud method has several advantages that make it efficient and useful in many applications.

It reveals information individually for each participant because it shows the interpretation of the interface for each participant. Thereby it is possible to understand the users' view on the system or product and thus recognize possible misconceptions. As people verbalize their thoughts it is possible to enrich a test report with vivid quotes by the participants. It is more tangible and convincing to read thoughts about problems literally than in paraphrased or aggregated state, e.g. in charts.

The Think Aloud method yields very qualitative data, even from a small number of participants. That means major problems of an application can be found and substantiated using the users' behavior and quotes.

In the concurrent version of the Think Aloud method the data collection takes place while the user task is executed. Thus, the user has no time for rationalizations revealing an unobstructed view on the results. The results reflect initial, candid and unfiltered reactions of the participants.

In contrast to other usability techniques the Think Aloud method enables the supervisor to detect even small irritations on the informal comments. Even if these irritations are not crucial to the overall success they can be fixed.

As the actions are visible on a very precise level it is possible to detect

deviations of the user actions from the ideal path. The supervisor of a usability study has the option to immediately respond to the procedure and thus dig deeper into a problem or to give necessary hints.

2.2.3 Disadvantages

Despite all the advantages, there are several disadvantages and situations when the Think Aloud method should not be the means of choice to conduct a usability study.

Conducting a Think Aloud study consumes more time and resources compared to other techniques, which possibly run automated and remotely. For many participants it is unnatural to verbalize thoughts, and encouraging them to do can make them feel insecure. It can be difficult for them to process information on multiple levels (see [Sea80] as reported by [BJLP02]).

If a study also collects performance data it has to be kept in mind that the Think Aloud method can have an impact on the performance metrics. This impact can have diverse extents and can either improve or decrease the participant's performance. Verbalizing thoughts is an additional mental operation and thereby inherently exhausting. This can decrease the performance of several participants. On the other hand it can happen that participants work on a task more consciously and thus their performance - despite the additional mental operation - is improved. Studies showed that the performance of participants could be increased:

In one case the performance was increased by 9% [BB84] In another case the participants had only 20% of the errors and performed twice as fast [WC92]. Nevertheless one has to keep in mind that the Think Aloud method when improving the participants' performance artificially overlays problems, that users would face when not working so consciously. That is particularly the case in repetitive tasks of power users that are performed rather unconsciously. Moreover, these power users often have more problems verbalizing their thoughts because they are not explicitly aware of their single actions.

Moreover, the environment established for the Think Aloud method can be a possible irritation to the participants. For instance, the presence of the supervisor or other observers can have an (aware or unaware) impact on the participant and thus bias the results.

2.2.4 Variations of the Approach

In addition to the originally proposed Think Aloud method, several variants of this can be useful, depending on the application.

As an example, the *Constructive Interaction* method [ODR84] encourages two participant to work on a task as a team while thinking aloud.

This feels more natural for the participants. As they talk to each other the interaction is more vivid, comments are made as a normal part of the interaction and not particular for the supervisor of the study [HB92].

The approach is useful in addition to the classical Think Aloud method. Relying solely on the results of the Constructive Interaction method can omit the fact that some participants might be able to cooperate better or worse than others. The method is particularly useful when conducting a usability study with children, as is feels much more natural for children to talk to other children than to a foreign supervisor.

Another constraint for the Constructive Interaction variation is that there must be many participants available as there is a double demand in comparison to the classical Think Aloud approach.

For further variations of the Think Aloud method see [RCS11, p. 293 ff].

2.2.5 Guidelines for Conducting a Study

A sample of guidelines worthwhile being mentioned is presented here. It is partially derived from the advantages and disadvantages mentioned above and partially adapted from [RCS11, p. 205 ff]

- Be impartial, i.e. react in the same way on mistakes and success
- Never make the participant feel incompetent
- Be aware of the voice and body language. High pitch voice reveals a positive reaction while a low voice can be regarded as a negative reaction
- Following the slogan "Quality over Quantity" it is better to test fewer participants but be mentally fresh than testing lots of participants but being exhausted
- Do not use the Think Aloud approach for tasks that are usually done unconsciously
- Pay attention when the participant is silent. Sometimes it reveals more information to note the incident and the participant's action rather than urging the participant to think aloud again
- Show interest in the participant's comments by sometimes repeating and affirming his or her comments
- Wait patiently
- Take care for participants' non-verbal signs of confusion and act accordingly
- Only work on one issue at one time and do not reveal information about other issues the participant has not discovered yet
- Do not help participants too quickly when they struggle with the application

- Hints should always be noted and only be given as last alternative, because:

  > "As soon as you assist, you are affecting the test results in a major way." [RCS11, p. 211 ff]

  This particularly holds for studies were several techniques are applied simultaneously.

For further information about when and how to give hints, see [RCS11, p. 212 ff]

After the usability technique to use is presented, the application they are applied to is introduced.

2.3 RCE - The Remote Component Environment

The Remote Component Environment (RCE) is a workflow-driven integration framework and serves as an example for the application of the analysis of usability problems.

This section gives a short introduction about RCE, containing general information, the system architecture of RCE (derived from [SFL+12]) and an introduction into the graphical user interface (GUI).

The GUI is of particular interest, because the solutions to identified usability problems will mainly be applied on GUI-level. Changes in the backend or the fundamental mechanisms of RCE are not applicable within the temporal restrictions of the thesis.

2.3.1 Origin and Current Application

RCE is an application to create and execute workflows consisting of workflow components. It is an integrated framework developed by the DLR facility Simulation and Software Technology. RCE allows the integration of different domain-specific tools from local or remote locations into a joined calculation. Thereby cooperation between experts and engineers from various domains at distributed sites is enabled. This is particularly useful when data and work of different domains strongly depend on each other. For example, during the design of an aircraft the dimensions of the wings, the design of the fuselage and the required power of the engine influence and depend on each other.

RCE was originally developed in the context of shipyard industry to design ships. Many components, functionalities and concepts are also derivable onto other contexts. Meanwhile, RCE is in use in other fields of research and development. For instance, in the early design phase of an aircraft, in the satellite design or in the modeling of the air traffic system RCE supports the cooperation of engineers.

The application RCE is open source and licensed under the *Eclipse Pub-*

lic License (EPL). Though the workflow components are mainly written by the RCE development team, it is generally possible that external workflow component writers contribute to RCE. Workflow components can be individually licensed. This is particularly useful, if components use closed source libraries. RCE is platform independent and currently in use under the operating systems Windows and Linux.

2.3.2 System Architecture

RCE is based on RCP, thus avoiding licensing issues and supporting the component-based approach.

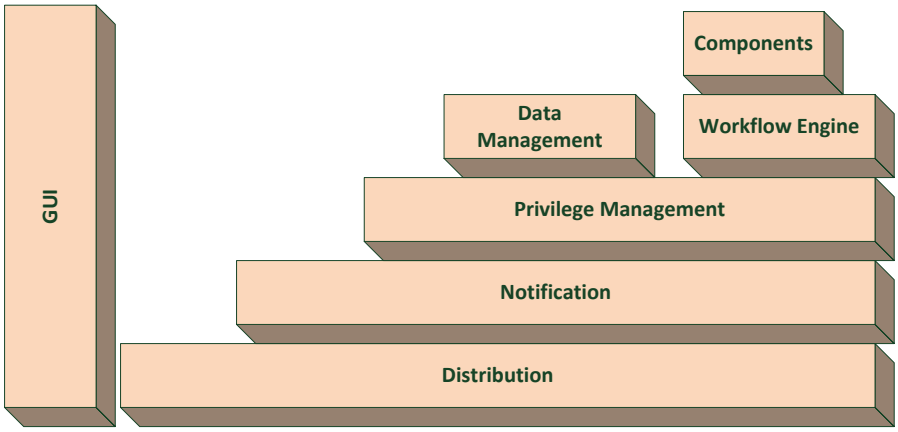

Figure 2.2: RCE software components and its dependencies

Figure 2.2 shows the RCE software components. The principle of this is approach is, that the upper layers can consume services from the lower ones.

The GUI components located next to all other components can consume services from them. Thus, functionality across various layers is possible, e.g. to depict states about the distributed system or to show notifications to the end user.

The GUI part is most crucial to the rest of the thesis.

The GUI elements are standard RCP components, enhanced with other RCE-specific parts, e.g. a login dialog, a log browser, a data management view or a graphical workflow editor. All parts are located in separate bundles. They can be added via the extension point mechanism (see [HS08]).

For example, the menu entry for the login dialog is added to the "File"-menu common in many applications. It is important that the GUI bundles can be deactivated, as some components are required to run headless, i.e. without a GUI.

For example, a component representing a database server that is used as remote service does not need a graphical interface.

2.3.3 GUI Parts

In this section the major parts of the GUI of RCE are presented. Figure 2.3 shows a prototypical state of the GUI when working with RCE.

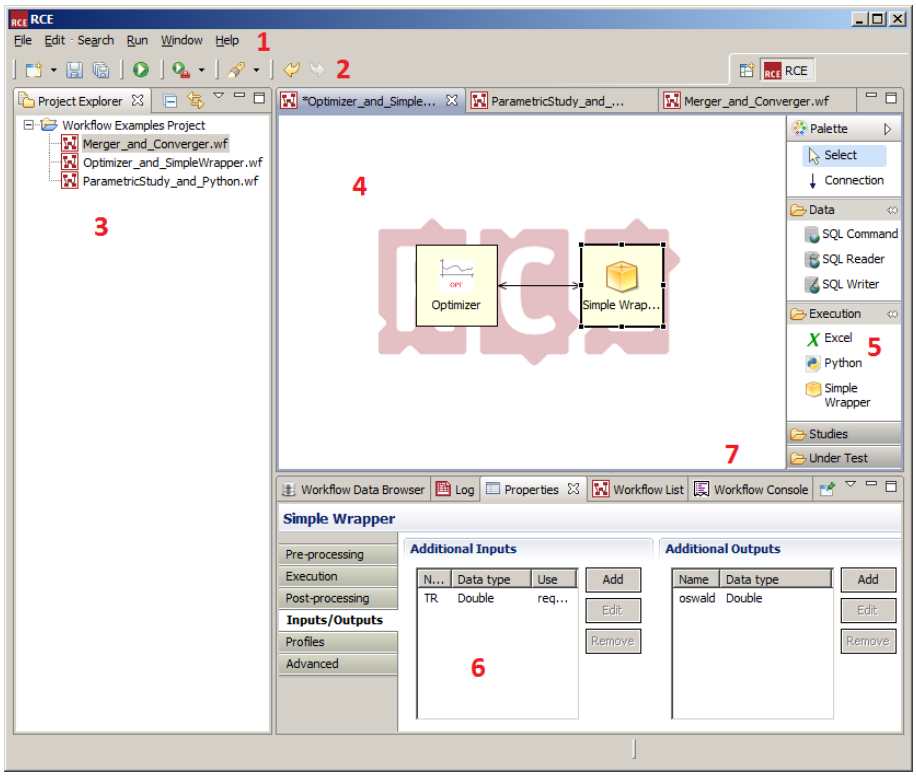

Figure 2.3: Screenshot of the graphical user interface of RCE

The numbered parts are described in the following list:

1. Menu Bar: The menu bar contains standard menus and entries known from other application, like saving files, opening files, editing and searching functionality
2. Tool Bar: The tool bar contains controls for the workflows, i.e. pausing, canceling, resuming
3. Project Explorer: The project explorer contains the projects and files (e.g. scripts or input data) located in the workspace folder in a tree view.
4. Graphical Workflow Editor: On the surface of the graphical workflow editor workflow components can be added, arranged and connected. A workflow component is represented by a rectangle holding an icon and a name.
5. Palette: The palette shows tools and a list of all components available

6. Properties: The properties tab contains configurations for each component, for instance the definition of inputs and outputs.
7. Others: At the same location the views for the workflow data browser, the log, a workflow list and a workflow console are placed.

The functionality to customize the application by placing different views at various positions in the GUI is part of RCP and also available in RCE. The screenshot presented in Figure 2.3 shows the default layout of RCE.

After RCE is introduced, the technical background influencing the implementation is of interest. Therefore an introduction to RCP, JFace and SWT is given.

2.4 RCP, JFace and SWT

To recall, usability problems and possible design improvements are expected on GUI-level. Thus, the relation of RCE to the technologies *JFace* and the *Standard Widget Toolkit* (SWT) [Ecl13] which mainly compose the graphical user interface is introduced here.

2.4.1 Benefits of the Eclipse Rich Client Platform

The main benefits of the Eclipse Rich Client Platform as presented in [MLA10, p. 5 ff] are:

Functionalities and code are encapsulated in components. These components are also called plug-ins. Components can be combined as desired, thus minimizing overhead by leaving out unnecessary components and maximizing flexibility by dynamically loading desired plug-ins providing additional functionality.

Moreover, as lot of infrastructure comes shipped with RCP, the focus is the domain and not on the infrastructure. Thus, developers can concentrate on how to enhance their views and not how to let them interact.

2.4.2 UI Toolkits JFace and SWT

In the context of the GUI of RCP applications the UI toolkits JFace and the SWT are important. The following sketch depicts their relations:

The following explanation describing Figure 2.4 is derived from [Rub06]. Eclipse RCP contains these parts:

- UI Workbench with editors, perspectives, and views.
- SWT, low-level graphics library of Java GUI components. It contains native implementations that hide differences between platforms or operating systems. Thus, there is a single API for the developer for all platforms.

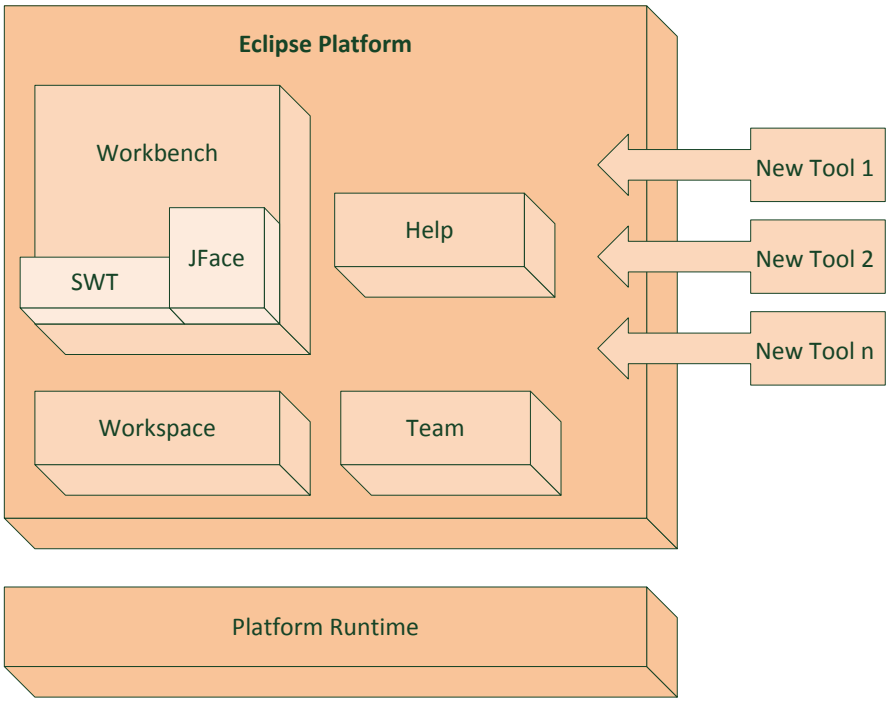

Figure 2.4: Overview of the Eclipse Development Environment [Rub06]

- JFace, a GUI abstraction layer for displaying objects that is layered on top of SWT. It helps developers by providing helper classes for features that can be tedious to implement.
- Platform Runtime, which defines a model to add extensions. Thus, loose coupling between plug-ins and just-in-time lazy loading and initialization is enabled.

The other parts are not crucial to the scope of the thesis and are not further explained here. [Rub06][IBM07]

To conclude the foundations the creativity method *Six Thinking Hats* is presented. It is used to gather redesign suggestion.

2.5 Creativity Method Six Thinking Hats

The *Six Thinking Hats* is a creativity method developed by Edward de Bono [DB99]. According to de Bono the concept is described as follows:

> " ... [A thinker] becomes able to separate emotion from logic, creativity from information, and so on. The concept is that of the six thinking hats." [DB99, p. xi]

The hat is a metaphor you can apply several expressions people are familiar with from real-life. It is a tangible object, thus virtually or literally putting on a specific hat means to think with a certain attitude.

This attitude is a deliberate process and each attitude is set exclusively. In the same manner, switching the hat symbolizes jointly switching the attitude towards the topic. [Car96].

De Bono describes the way the hats influence a discussion as follows:

> "The six thinking hats allow us to conduct our thinking as a conductor might lead an orchestra." [DB99, p. xii]

2.5.1 Six Colors - Six Attitudes

There are six hats, each one associated with a color and an attitude towards the topic under observation (see Figure 2.5).

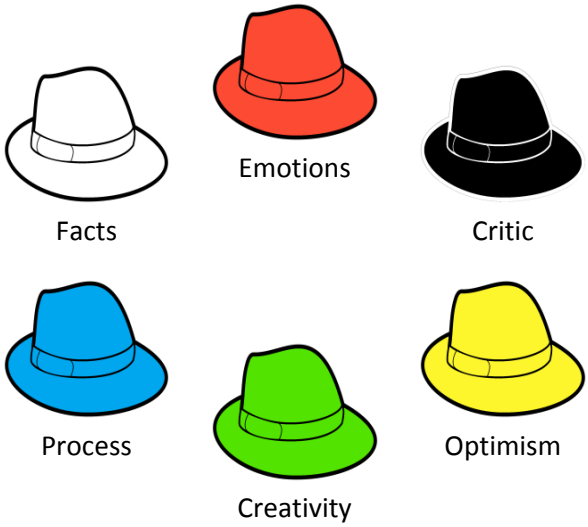

Figure 2.5: Colors and attitudes of the Six Thinking Hats

The idea behind the hats as presented in [DB99, p. 25 ff] is explained here:

- White hat - Facts: The white hat is associated with a neutral and objective attitude. It takes care of facts, numbers and information.
- Red Hat - Emotions: The red hat represents emotional attitudes like wrath, fury, intuition. Hunches and gut feelings are also covered by that hat.
- Black Hat - Critic: The black hat simulates a sad and negative attitude towards the issue under observation. Negative aspects, possible risks and other problems that could arise from the issue are in the focus of this hat.
- Yellow Hat - Optimism: The opposite of the black hat is the yellow one. It is associated with positive and glad feelings. Optimism, hope and possible benefits that might arise from the issue are the aim of this hat.

- Green Hat - Creativity: The green hat is associated with creativity. Symbolizing vegetation and fertile growth the green hat is used to generate new creative ideas, assuming anything is possible.
- Blue Hat - Process: The blue hat is a meta hat. It is associated with the organization of the process itself. Thus, the agenda, the process and the desired decisions are determined while this hat is put on.

2.5.2 Benefits of the Method

The creativity method Six Thinking Hats has several benefits.

Within the process of the method all members of the team are free to speak out their ideas without any risk. Several points of view towards the issue are examined and thus the awareness towards the issue is enhanced. Moreover the here describes method has become a convenient mechanism for thinking in new ways applying simple rules. These rules help the team to focus their thinking onto one specific attitude at a time. Several orders of the hats are proposed (see [Ser12]) depending on the application.

It is assumed that this leads to more creativity in the thinking of the members of the team. As the method is very verbose, the communication within the team is improved. The process of decision making is improved since a lot of aspects are stated, weighted and discussed.

After the foundations for the thesis are clarified, the conception is presented in the next chapter.

Chapter 3

Conception

"There is nothing worse than a sharp image of a fuzzy concept."

—Ansel Adams

The conception can be divided into the presentation of the general strategy, the conception of the two techniques to be compared (Track-Analysis-Tool and Think Aloud method) and the metrics to be measured.

3.1 Strategy

As described in the problem definition, the thesis aims to develop a tool to analyze usability problems in depth and to compare the results with an existing usability technique as reference.

This chapter describes the strategy and the concepts required to accomplish this.

First, the *Track-Analysis-Tool* developed in the course of the thesis is designed. The Track-Analysis-Tool consists of two parts. One part that makes sure that all required interaction of the user with the interface is captured and another part that executes the analysis given additional knowledge. Therefore technical terms vital to the approach and the developed algorithms are described.

Second, the selection of a reference usability technique is presented. Requirements for the technique are stated and a compilation of existing technique is created. Eventually reasons for the Think Aloud method as chosen reference technique are presented.

To allow for comparison of the results, usability metrics are defined. On the one hand a set of standardized metrics is selected. On the other

hand application specific metrics are defined to enhance the approach of usability problem analysis.

3.2 Track-Analysis-Tool

As mentioned above, the Track-Analysis-Tool consists of two parts:

- User interaction with the interface is captured
- Captured user interactions are analyzed

The capture of interactions is realized on the example of the workflow-driven integration environment RCE. It is assumed that a sufficient amount and variety of interactions of the user with the interface can be captured.

The approach regarding the analysis is as follows: To recall, the abstract usability problem under observation is that people struggle with RCE when they use it for the first time. Derived from that a user task should cover the first starting of RCE up to the first successful execution of an exemplary workflow.

This section describes the concept of subtasks as segments of the user task that can be examined separately. Moreover the meaning of interaction patterns in the context of the thesis are described. As an enhancement of the analysis, mouse track visualizations are explored. An abstract algorithm to assign interactions to their respective subtask and a further abstract algorithm to rate a subtask given certain additional contextual knowledge are presented.

3.2.1 Definition of a Subtask

In order to understand the relation between a task, a subtask and interactions it has to be described in more detail, what the term *Subtask* means.

For short, a subtask is a subunit or a fraction of a task. A task is for example a user task given to a user in the course of a study. On the example of RCE the overall task is to create, configure and execute a first workflow. The task possibly has a defined goal, but can also be formulated more freely. Accordingly, a subtask can also have a fixed goal. An example for a subtask can be the insertion of workflow components. The respective goal is reached when all required workflow components are added to the workflow.

Referring to the context of the thesis, a subtask is a division of the user task the probands ought to fulfill in the study. Each subtask is defined by a name and a distinct beginning and end. The beginning and end can be a certain event or state of the system or a combination of these. Subtasks can be processed sequentially or concurrently.

In the literature other terms for related constructs or concepts are used, depending on the domain. Alternative terms are context, task, process, activity, activity cluster, task-related subsection or from an abstract point of view: knowledge.

In the context of the thesis the term subtask is understood as described above.

3.2.2 Subtask Segmentation

Automated subtask segmentation is desirable, as is makes the tool applicable on a more universal scope. This is particularly useful when the task to fulfill is very complex and there are various ways to fulfill it. Thus, an automated approach would generically map these ways of fulfillment as a sequence of fulfilled subtasks to the results. It yields clues about the way people interact with a certain software and which solution strategies are applied.

Literature proposes several approaches to that issue (see [GRA10][LFBG07][LG08]). The related work shows that it is generally possible to detect subtasks to some extent, but it is a complex task, that still needs manually defined training data, e.g. when applying machine learning techniques (see [LFBG07]).

Moreover, the results can be imprecise. In [CS09b] a study testing an segmentation approach was conducted. The correctly detected tasks were averagely 81%, ranging from 100% to 66%. In the study conducted in the course of this thesis only a small number of probands is tested. Thus, a deviation in correctly detected tasks has significant impact.

To conclude, the drawbacks of an automated approach mentioned above are compared with the additional effort a manual definition of subtasks would cause:

Given a large number of subtasks, the manual definition requires a lot of effort and is error prone. In this case the effort to integrate a generic approach would pay off. As the number of subtasks is quite small, only one single task is examined and they are easily distinguishable by human beings they are defined manually.

To recall, the overall task is to create, configure and execute a first workflow successfully. The following list gives an overview about the subtasks that the overall task has been devided into:

1. Start the "New Project" wizard
2. Execute the "New Project" wizard
3. Start the "New Workflow" wizard
4. Execute the "New Workflow" wizard
5. Open the workflow

6. Add the required components
7. Configure the components
8. Configure required connections between components
9. Start the "Execute Workflow" wizard
10. Execute the "Execute Workflow" wizard

A vital part for the manual definition of subtasks is a distinct beginning and end. The subtasks are required to be executed subsequently. Concurrent subtasks would increase the complexity tremendously and are thus excluded from consideration. If subtasks can run concurrently the assignment has to be done in a more sophisticated way. Given the points of beginning and end of a subtask are correctly detected, the events can be assigned accordingly.

A typical beginning of a subtask is the start of a wizard or the opening of the graphical workflow editor. A list of the beginnings and ends for the subtasks mentioned above can be found in Appendix A—"Beginning and End Conditions of the Subtasks".

The identification of the beginning and end of a subtask enables the assignment of patterns to subtasks. Moreover, the duration and a mouse track for each subtask can be associated.

The granularity of the subtasks is further elaborated and explored. It may be the case that the cause of a usability problem is on a more abstract level than represented by the subtasks.

For example, the analysis might yield the result that a wizard must be redesigned, although the actual problem is because a wizard might not fit in that specific situation. So an optimization of the wizard would run into a local optimum but would fail to optimize on the global scope. In this case, further examinations are required.

3.2.3 Interaction Patterns

Interaction patterns can be interpreted as interactions between the graphical user interface and the user on several levels of abstraction. For example, the starting of a wizard might be seen as a pattern or a simple click on a button. To proceed towards usability problems ratings are assigned to each pattern depending on whether it contributes to solving a particular subtask or not.

Patterns on a higher level of abstraction can be seen as a combination of the simple interaction patterns. On this level of abstraction there are lots of general patterns common in many applications. Others might be entirely context specific and must be defined separately.

The following list shows a sample of interaction patterns on a higher level of abstraction:

- Filling out a form
- Choosing values from a drop down menu
- Fulfilling a wizard
- Move elements on a graphical surface
- Find an element in a tree viewer
- Operate the search function
- Writing an email

Moreover patterns on the lowest level are investigated. Therefore a sample of common GUI elements and their typical way of interaction is collected.

A sample of significant patterns on the lowest level is presented here. For a more exhaustive list, see Appendix B—"List of Typical GUI Elements and Interaction Patterns".

- Button: Select
- Checkbox: Check, uncheck
- Wizard: Open, back, next, cancel, finish, close
- Menu: Show, hide
- Palette item: Drag, drop, select

3.2.4 Mouse Track

The mouse positions per subtask are captured. An analysis based on existing approaches with mouse tracks might help to understand the identified usability problem in more depth.

One way to capture a mouse track is the use of an external tool. A basic test showed that *IOGraph* [ZS13] serves for the purpose of capturing mouse tracks, but lacks the integration into RCE and thus makes controls complicated.

As described in 2.4.2—"UI Toolkits JFace and SWT" RCE uses SWT as framework for the graphical user interface. A first test showed that it is generally possible to automatically collect the mouse movement from all views of interest. The resulting mouse movement was rudimentary visualized using SWT graphics functionality.

There are various more sophisticated visualization techniques imaginable. For instance, Arroyo et al. [ASW06] present an approach how to visualize mouse tracks. Moreover, the statistics application *R* [Adl10] can be used to visualize two dimensional data as a heatmap. Heatmaps are often used to visualize the attention of users on websites [OS08][AL08][HNZ]. An example for a heatmap can be seen in Figure 3.1.

Figure 3.1: Exemplary heatmaps visualizing eyetracking data [Nie06]

3.2.5 Analysis Algorithms

In this section the algorithms for subtask detection and subtask rating are presented. Correctness, completeness and complexity are examined and examples are given.

Algorithm for Subtask Detection

In order to assign the events to their respective subtask an algorithm was designed:

Data:

- Set of events $\mathbb{E} = \{e_1, e_2, ..., e_n\}$
- Per subtask with identifier n: A set of conditions to fulfill $\mathbb{C}_n = \{c_{n1}, c_{n2}, ..., c_{nm}\}$

Result:

- Per subtask with identifier n: A set of events $\mathbb{E}_n = \{e_{n1}, e_{n2}, ..., e_{nm}\}$

```
1  x = 1
2  create priority queue P_e containing all events of E ordered by
   time stamp
3  forall the events e in P_e do
4      add e to E_x
5      if e fulfills a condition c_nm then
6          mark c_nm as fulfilled
7          if all c_nm in a C_n are fulfilled then
8              mark C_n as fulfilled
9              x = n + 1
10         end
11     end
12 end
```

Figure 3.2: Subtask detection algorithm

The input of algorithm 3.2—"Algorithm for Subtask Detection" consists of two sets. Firstly, the set of events identified from the log, which in turn comes from the capture plug-in of the application. Secondly, per subtask a list of conditions that have to be fulfilled is defined.

The output of the algorithm is one list of events per subtask.

The algorithm works as follows:

The assignment index x is initialized as 1. I.e. the algorithm starts to assign events to the subtask with the ID 1. Then a priority queue containing all events of the set $\mathbb{E}$ is created, using the time stamp as ordering key. This enables the algorithm to process logs containing events that are not necessarily chronologically ordered or combined logs from various sources.

The events are processed in the order of their time stamps. That means the earliest occurring event is processed at first, while the rest follows sequentially. Thus, it is assumed that all events that do not cause a change of the subtask can be assigned to the currently active subtask. As a first step, the event under observation is added to the currently active subtask. That means, if an event completes a subtask, it is assigned to this subtask and not to the next one.

For each event in the input set it is checked whether it fulfills one of the conditions c_{nm} for one of the subtasks, as defined in $\mathbb{C}_n$. If a condition is fulfilled, it is marked so. If then all conditions in $\mathbb{C}_n$ for the subtask with the identifier n are fulfilled, the set of conditions $\mathbb{C}_n$ and thus the subtask is marked as fulfilled. The assignment index x is set to the identifier of the just fulfilled subtask incremented by 1. Thus, the subsequent events are assigned to the next subtask, until another subtask is marked as fulfilled.

The time complexity of the algorithm can be determined line by line: Setting the index has constant time complexity (line 1). The creation of the priority queue has a time complexity of $\mathcal{O}(n * log(n))$. The enqueue operation (complexity of $\mathcal{O}(log(n))$) is executed n times (line 2). The loop is executed n times, once for each event (line 3). Checking the condition of the first conditional statement in line 5 is linear in the size of all $\mathbb{C}_n$, i.e. the number of all fulfillment conditions. Additionally, checking the condition of the second of statement in line 7 is linear in the size of the respective $\mathbb{C}_n$, i.e. the conditions for the subtask potentially being fulfilled. The other actions within the for loop have linear time complexity.

Thus, the complexity of the for loop adds up to $\mathcal{O}(n*m)$, where m is the amount of all conditions $|\mathbb{C}_1| + |\mathbb{C}_2| + ... + |\mathbb{C}_n|$. The overall algorithm has a complexity of $\mathcal{O}(n * log(n) + n * m) = \mathcal{O}(n * m)$, where m is $|\mathbb{C}_1| + |\mathbb{C}_2| + ... + |\mathbb{C}_n|$ and n is the number of events to be processed.

Concrete example

As a concrete example one might think about a subtask where several configurations have to be done in order to fulfill the subtask. From these configurations it is possible to derive conditions. For example, a choice in a drop-down menu, inputs to text fields or the selection of a radio button might be required. In addition to that, more application specific events on a different level of abstraction can be seen conditions, for example the definition of connections or adding elements to a canvas.

The algorithm iterates over the ordered events. When an event indicating a valid selection in the respective drop-down menu or radio button is identified, the respective condition is marked as fulfilled. When the fulfillment of a condition can be revoked later on, e.g. by deleting text from a text field after successful insertion, the mark is removed. For sake of simplicity, the deletion of marks is omitted in the algorithm. Whenever a condition is met, the subtask is verified to ensure all conditions are met. In this case, when a radio button is pressed, the condition of selecting an item in the drop-down menu and entering texts into the fields are verified to ensure they are fulfilled. The same holds for the other conditions. Assuming that the selection of a radio button was the last condition that had to be fulfilled, the set of conditions for this subtask is marked as fulfilled. The index for the next event under observation is incremented by one. The event that caused the fulfillment of the subtask (i.e. the selection of the respective radio button) was already assigned to the subtask it contributed to.

Algorithm for Subtask Rating

After the events have been assigned to their subtasks successfully, the subtasks are rated. Therefore the following algorithm is defined:

Data:

- Threshold factor t
- Per subtask with identifier n:
  - A set of events $E_n = \{e_{n1}, e_{n2}, ..., e_{nm}\}$
  - A set of ratings $R_n = \{r_{n1}, r_{n2}, ..., r_{nm}\}$
  - A set of reference values V_n

Result:

- A set of rated subtasks $\mathbb{S} = \{s_1, s_2, ..., s_n\}$

```
1  foreach subtask with identifier n do
2      c_g, c_pos, c_neg, c_neut = 0
3      forall the events e in E_n do
4          find rating for e_nm in corresponding R_n
5          set rating for e_nm accordingly
6          increment respective counter c_pos, c_neg or c_neut
7      end
8      determine duration of subtask s_n using E_n
9      determine confusion flag using E_n, t and V_n
10     add reference duration from V_n to s_n
11     add actual duration to s_n
12     add confusion flag to s_n
13     add overall number of events |E_n| to s_n
14     add actual counters c_pos, c_neg and c_neut to s_n
15     add reference counters from V_n to s_n
16     add s_n to S
17 end
```

Figure 3.3: Subtask rating algorithm

Algorithm 3.3—"Algorithm for Subtask Rating" has several inputs. The threshold factor t determines the sensitivity of the algorithm to generate confusion warnings. Depending on the domain this factor can be used to tune the accuracy of the analysis. Moreover three lists per subtask are taken as input.

- E_n contains all events assigned to the subtask with identifier n.
- R_n contains ratings for specific events in the context of the respective subtask. For example, entering text into a specific text field might contribute to subtask A but is not useful for subtask B and thus a hint for a potential usability problem in the context of subtask B.
- V_n contains some reference values that give an indication how long a subtask should take. If the actual value exceeds the reference value to a high extent it is a hint for possible usability problem. The reference values per subtask are: The duration, the amount of negative events, the amount of neutral events, the

amount of positive events and the amount casual events.

The output of the algorithm is a set of rated subtasks $\mathbb{S}$, where each subtask s_n holds several properties.

For sake of readability a listing of the reference values in V_n and the properties per subtask s_n in the algorithm is omitted.

The properties that are added to each subtask are the following:

- The actual duration and a reference duration
- The confusion flag
- The overall amount of events
- The actual amount of positive, neutral and negative events
- The reference values for positive, neutral and negative events

Though there are approaches [SK05b] to combine various aspects into one single number as a metric, the approach presented here leaves the interpretation of the properties per subtask to a usability expert analyzing the outcome.

It is assumed that it is a non-trivial task to combine this information into one single number in a robust manner. Thus, the values are added to the subtasks and the task of interpreting these values is left to the usability expert analyzing the outcome.

The algorithm rates the subtasks using the following procedure:

Per subtask the list of events that has been assigned to the subtask is processed. Each event is rated as either positive, negative or neutral using the list of ratings R_n for that specific subtask with identifier n. As mentioned above the ratings are subtask specific, that means an event can be regarded as positive or negative, depending on its context. For example, opening a menu can be positive, if an entry required to fulfill the subtask is contained in this menu. If the same menu is opened in another context where it is not expedient, the event is rated differently. Depending on the rating the respective counter is incremented. The duration of the subtask is determined as the difference of the extremes of the occurring time stamps in the events of a subtask.

To determine whether the confusion flag is set to true or false an approach using E_n, t and a reference value r from V_n is applied. First, the number of so-called casual events is counted as c_{temp}. Casual events are events that are often invoked unintentionally, for example focus events, mouse events or arm events. If that number is higher than the product of a reference value r from V_n and the threshold factor t, it is assumed that the events have been triggered due to confusion and thus the confusion flag is set to true. For short: if $c_{temp} > t * r$ the confusion flag is set to true.

In lines 9-14 the respective variables are set in the rated subtask. Finally, the rated subtask s_n is added to the set of subtasks S.

The time complexity can be determined by analyzing the algorithm's loops, as all operations inside the loops have constant time complexity. The inner for loop (line 3) is iterated over all event lists (line 1) thus overall each event is processed exactly once. So the time complexity is $\mathcal{O}(n * c) = \mathcal{O}(n)$, where n is the overall number of events and c is the constant procession time per event. To sum up, the algorithm has linear time complexity in the number of events.

Concrete example

A concrete example for the rating algorithm might be the set of events assigned to the subtask of opening a specific dialog. In the course of fulfilling this subtask the user is looking for a way to open this dialog, might browse several menus (and invokes lots of arm events), open a different dialog at first, cancel this dialog, browse further menus until the required menu entry is found and the desired dialog is opened.

From this sample of events, canceling the first dialog would be regarded as a negative event, opening the correct menu and selecting the required menu entry would be regarded as a positive event and the other events are neutral. Depending on the given threshold t the number of invoked casual events, particularly the arm events, exceeds the product of reference value and threshold and thus the confusion flag is set to true.

The subtask is rated and the fact that there was a negative event and that the confusion flag indicated confusion it is likely that this subtask is further examined by the analyst.

Possible adaptation

A possible adaptation of this algorithm would be to use separate thresholds for each subtask. In the course of the thesis is it assumed that this approach is too prone to overfitting and thus not applied here. Future work might examine this issue in more depth.

The Track-Analysis-Tool is compared with another usability technique. The determination of this reference technique is presented next.

3.3 Reference Usability Technique

In this section the requirements for a reference usability technique are stated. In order to find a suitable usability technique to serve as a reference to the Track-Analysis-Tool described above, several techniques proposed in the literature are compared. This section concludes with reasons for the selection of the respective method. Due to the temporal restriction of the thesis the selection of the technique is solely based on

literature.

3.3.1 Requirements for the Reference Technique

To be able to select a reference technique it is important to clarify the requirements. As the reference technique is supposed to be applied within the temporal constraints of the thesis, the required amount of time is supposed to be small. For the same reason the required amount of users should be small, as the acquisition of probands takes a certain amount of time. As neither the author of the thesis nor anyone of the RCE development team is a usability expert the expertise of a possible evaluator should be minimal.

Another property that would enhance the comparison of the two approaches is a common time axis. Particularly in the development phase it will help to compare the findings.

The selected reference technique must be able to detect major usability problems. When less critical usability problems are detected it is no problem, but the focus is on crucial problems. As the application RCE is already implemented and in use (but still evolving), a formative study has to be conducted. Thus, the reference technique must be applicable on existing products. Techniques only suitable for the design phase or for finalized products are not fitting here.

Moreover one should keep in mind that the target group of the thesis is novice users, regarding their experience with RCE.

3.3.2 Comparison of Existing Techniques

There are several comparisons of existing usability techniques presented in the literature. In this section the comparisons presented by Nielsen [Nie94b, p. 224], Holzinger [Hol05] and Sarodnick [SB06] are examined. The combined properties of the techniques are shown in Table 3.1—"Usability techniques: Overview of inspection methods" and Table 3.2—"Usability techniques: Overview of test methods".

The techniques are classified into inspection methods and test methods. When applying inspection methods, usability experts inspect the system. Real users are not required. Guidelines, GOMS[1], Heuristic Evaluation and Cognitive Walkthrough are assigned to this class of techniques. In contrast to these methods, the test methods involve actual users of the product. Eyetracking, Think Aloud Method, Questionnaires and Field Observation are located in this group.

Most of the properties are self-explanatory and do not need further explanation. For some of them further information is given here:

[1] GOMS stands for **G**oals, **O**perators, **M**ethods and **S**election rules. It is a model to analyze human-computer interaction.

	Inspection Methods			
	Guidelines	GOMS	Heuristic Evaluation	Cognitive Walkthrough
Probands	none	none	none	none
Evaluator's Expertise	+	+++	++	+++
Evaluators			3+	3+
Amount of Time	+++	+++	+	++
Phase			all/early design	all
Equipment	+	+	+	++
Intrusive	no	no	no	no
Productivity	+++	+	+++	++
Specificity	+++	+++	+++	++
Common Time Axis	no	no	no	no

Table 3.1: Usability techniques: Overview of inspection methods

	Test Methods			
	Eyetracking	Think Aloud Method	Questionnaires	Field Observation
Probands	5+	3+	30+	3+
Evaluator's Expertise		+	+	++
Evaluators	1	1	1	1+
Amount of Time	+++	+++	+	+++
Phase		all/design	all/follow-up	follow-up
Equipment	+++	+++	+	++
Intrusive	no	yes	no	yes
Productivity		+++	+	+++
Specificity		+++	+	+++
Common Time Axis	no	yes	no	yes

Table 3.2: Usability techniques: Overview of test methods

- Phase in Product Lifecycle: The assignments are stated in the literature are rather imprecise as they are based on different lifecycle approaches (iterative vs. linear). Nevertheless, the stated phases are assumed to give a feeling about suitable techniques.
- Required Equipment: The required equipment subsumes all purchases that have to be done to conduct the study successfully, e.g. software or a video recorder.
- Intrusive: A technique is intrusive if it affects the user's work with the application.
- Productivity: Productivity represents the ratio of valuable information and the required time. If a method detects lots of usability problems within a short amount of time, the productivity will be rated "+++".
- Specificity: Specificity describes the level of abstraction of the problems found with the techniques. For instance, when there

is a problem filling out a form, some methods just yield the information that there is a problem with the form while other methods with a higher specificity yield information which problem users had with which element of the form.

Some properties presented in the original sources which are not crucial for the selection were omitted. In some cases the tables used in the literature are blank. In some cases reasonable values could be entered. In other cases the cells are left empty.

3.3.3 Reasons for the Think Aloud Method

An examination of the presented usability techniques let to the conclusion that the Think Aloud method is best suitable a reference technique for the context of the thesis.

There are some aspects why other techniques do not fit: For instance, Heuristic Evaluation and Cognitive Walkthrough require at least three evaluators. They are not available. Moreover, the results of the test methods are can be more tangible which is another drawback for the inspection methods.

On the other hand there are several arguments that substantiate the choice of the Think Aloud method:

A common time axis is an important property of the reference technique, which is only given for the Think Aloud method and Field Observation. Moreover, the productivity of the Think Aloud method is high. That means within a few conducted tests lot of information is gained. Similarly, the aspect of specificity is high when using the Think Aloud method. This is particularly important because the comparison with the Track-Analysis-Tool can just be as specific as the reference results obtained by the selected method.

The amount of users which are potential users of RCE but have no used RCE before is rather small. This and temporal consideration make the low amount of users needed for the Think Aloud method an important argument.

A final argument that substantiates the suitability of the Think Aloud method is the phase in the lifecycle when the technique can be applied. The Think Aloud method can be applied in the design phase, possibly as part of an iterative approach. As this meets the intended schedule of both the general RCE development and the study for the thesis, and thus fits very well.

After the conception of the Track-Analysis-Tool and the determination of the reference technique suitable metrics to allow for a comparison of the two methodologies are presented.

3.4 Usability Metrics

The selection of standardized usability metrics and additional application specific aspects which can be useful to determine usability problems are presented in this section.

3.4.1 Between vs. Within Comparisons

The usability study conducted in the course of the thesis has temporal constraints. So there is only one task designed. Thus, it is not possible to do within-subject testing. The testing can be seen as between-subject testing. [Nie94b, p. 179]

It is required to let each user just participate in one round to avoid learning effects. As only one task needs to be solved no learning effects can occur and no counterbalancing (see [TA10, p. 19]) is needed.

Within one iteration a comparison between probands can detect outliers in the metrics, which might be a hint for problems only faced by a few users. But this might also be caused by personal problems or misunderstandings. If the metrics of all probands are high it can be caused either because all users face a certain problem or because a subtask inherently takes longer than others. To give the usability expert who analyses the metrics reference values for the metrics are added.

3.4.2 Standardized Metrics

There are many metrics proposed in literature. A short overview and introduction on the topic is given in 2.1.3—"Metrics". The essence is: Metrics are a means to quantify properties of a thing or procedure.

This section describes which usability metrics are measured in the user study of the thesis and why.

To recall: In the context of the thesis improving usability is seen as the maximization of the three parts effectiveness, efficiency and satisfaction. Thus, the selection of the metrics strives to cover these three aspects.

Another requirement towards the metrics is that they are supposed to be simple and proven useful. Thus, metrics that are frequently used in other contexts are preferred.

Sauro and Lewis [SL09] examined which metrics are used frequently. There analysis is based on 97 data sets containing data of 2286 distinct users and distinct 1034 tasks. The frequencies of the metrics is shown in the following table by Sauro and Lewis [SL09]:

The paper examines the metrics and correlations between them. The main message can already be extracted when mapping the table to

	N	%
Task Time	96	99
Completion Rate	95	98
Errors	56	58
Post-Test Satisfaction	47	48
Post-Task Satisfaction	39	40

Table 3.3: Frequencies of metrics in user studies [SL09]

the three areas of focus mentioned above: Task time represents efficiency, completion rate and errors represent effectiveness and satisfaction speaks for itself.

In the following the metrics are surveyed in more detail:

Completion Rate

The completion rate is measured binary, that means the task is either accomplished entirely or not at all. This is the case because a halfway constructed workflow or a workflow that never ran to check its functionality is not likely to be useful. The task is weighted as success when the workflow runs correctly. When the workflow runs technically, but yields wrong results the user is allowed to retry. Thereby it is assumed that a deeper insight how users work can be gained. If the user gives up, the task is rated unsuccessful.

Errors

Errors in this context are seen synonymous as blocking usability problems that require help. Slight usability problems that did not require the intervention of the supervisor are not included here. Later on the term "hint" describes the same aspect as it is the immediate result from the error. Note that this error does not refer to error messages from the interface.

Task Time

When measuring the task time one has to keep in mind that it can interfere with the Think Aloud method. When people think aloud they can possibly just work slower because of the additional cognitive workload. But also the opposite can be case: Users might work faster because they are more concentrated on the task [RCS11, p. 80]. It is assumed that the influence is not significant.

In many cases it can be assumed that faster means better. There are some exceptions, for instance games, where the user wants to enjoy the gaming experience as long a possible or in learning applications, for the longer someone struggles with an issue the deeper it is anchored in mind. Moreover, anything else that serves as a pastime is not better

when it's quicker. In the context of the thesis faster is better. The overall task time is measured as well as the time per subtask.

Satisfaction

The table above distinguishes between post task and post test satisfaction. As the test of the user study conducted in the context of the thesis consists of just one task, it is same in this case. Satisfaction is a self-reported metric, i.e. it is reported by the users themselves and inherently subjective.

Questionnaires are a common technique to measure self-reported metrics like satisfaction. There are lots of standards proposed [Bro96][CDN88][Lew95][Lun01] containing various sets of questions. As the questions of the *Computer Usability Satisfaction Questionnaires* by Lewis (see [Lew95]) offer a view onto satisfaction from several perspectives it is chosen as the basis for the questionnaire.

The questionnaire uses a *Likert* scale (see [Lik32]) having 7 possible ratings per question. The number is intentionally odd since it allows users to express a neutral attitude. To the left and the right of the scale a statement representing the associated attitude is located.

Minimal and Maximal Values

It has to be considered whether expected values should be added to the analysis. For example, a maximal task time could be set and if it is exceeded the task is rated unsuccessful. Sauro and Kindlund [SK05a] propose a method to determine minimal and maximal values for the metrics, e.g. a maximal task time.

As the study is explorative and there is no knowledge about what to expect, such values are not determined. Despite that, a reference value how long an experienced users take for the different subtasks is given to detect subtasks that take significantly longer.

3.4.3 Application Specific Metrics

In addition to the standardized metrics further aspects are considered to allow for deeper insights.

Optimal Value Comparison

As reference values an average time an experienced user needs to fulfill a task, as well as the amount of neutral events is given. If the times are much higher for all users this might be a hint for a usability problem. There are subtasks that have a high learning curve, i.e. the first execution takes rather long but once the user has understood the principle the performance improves significantly. Though novice users will not compete with experienced users in terms of performance anyways too

low metrics could be seen as a hint to improve the subtask anyways. When the performance is regarded as "too low" is up to the analyst.

Confusion Factor

In addition to the negative events described above and long times for a subtask another hint for usability problems was integrated. Confusion is assumed when the amount of possibly unintentional events exceeds a reference value by a certain multiplication factor (see 3.3—"Algorithm for Subtask Rating"). To recall, unintentional events are focus events, mouse events and arm events. They are often fired unintentionally but rather as a follow-up of another event. For example, when a view is opened the focus changed several times. If this view is not needed to fulfill the task these focus events are not needed and give a hint about a possible usability problem.

Another good example for confusion is when lots of arm events are invoked. When users search for certain functionality they tend to browse through the menu entries. But only considering one of these types tends to be too sensitive. To depict this figure Figure 3.4 shows two ways a user can select a menu entry.

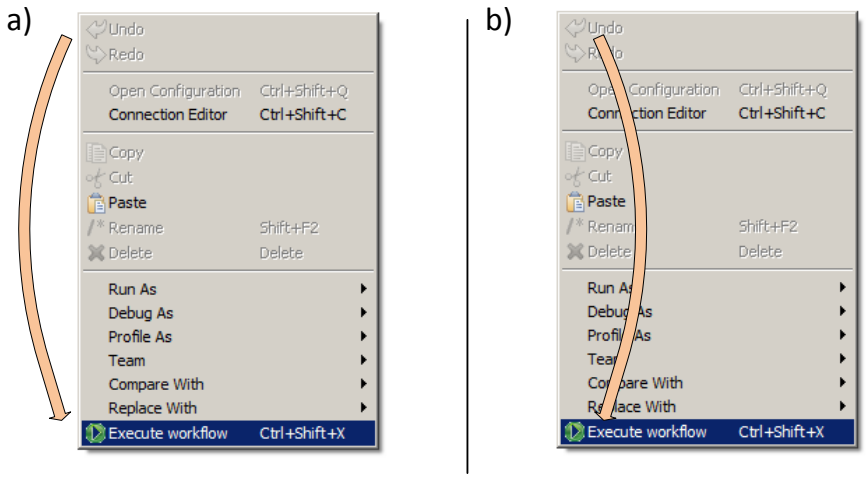

Figure 3.4: The entry "Execute Workflow" can be selected using different mouse paths

On the one hand, the user can move the mouse next to the menu and select the menu entry without moving over the other menu entries and on the other hand the user moves the mouse over all menu entries, invoking lots of arm events. The logged events significantly differ, though the semantic meaning is the same.

An issue open for future work is the determination of the multiplication factor. In the scope of the thesis the factor will be set to match the observed results. Of course, this approach is prone to over-fitting. To make this factor usable on a universal scale it can be determined by

empiric studies.

Distance from Success Vector

To provide a deeper understanding of the users' state of mind and their attempts to solve a subtask it is desirable to detect sequences of actions that almost solved a subtask but failed shortly before succeeding. Therefore a metric called *Distance from Success Vector* is introduced.

This approach can be suitable when there are several sequences of actions (hereafter called "path") that can solve a subtask. The set of paths form the Distance from Success Vector. The paths can have different lengths, thus both relative and absolute advance can be of interest. Additionally, a counter indicating how often a path was started can be integrated.

Figure 3.5 depicts the idea behind that:

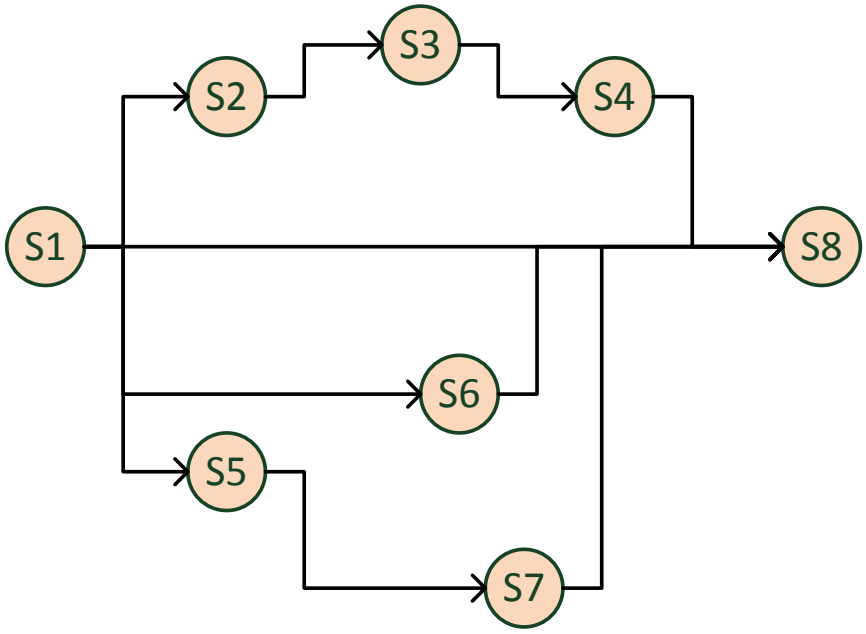

Figure 3.5: Depiction of a Distance to Success Vector

The nodes are states of the system and the edges are actions that cause the transition to the next state. Actions like closing a wizard or hiding a menu discard advances and set the system state respectively.

To exemplify this the subtask of opening the workflow wizard is further examined here. The workflow wizard can be started by various sequences having lengths between one and five actions. Table 3.4—"Sequences of actions for a Distance to Success Vector" shows prototypical paths.

Sequence of Actions	Length
Toolbar entry	1
Context menu - menu entry	2
File - menu entry	2
File - new - menu entry	3
File - new - other - rce - menu entry	5
Toolbar chevron - menu entry	2
Toolbar chevron - other - rce - menu entry	4
Toolbar chevron - rce - menu entry	3

Table 3.4: Sequences of actions for a Distance to Success Vector

To depict how a possible output of the Distance from Success Vector might look like the following behavior of the user can be imagined. At first, the user browses the context menu of the project explorer but does not find the menu entry though it is present. Then the user might open the general wizard via the chevron in the toolbar, click on "other", then on "RCE" but is not sure whether the entries visible there are the correct ones. So the user cancels the wizard and continues searching. The next location under observation is the "file" entry in the menu bar. The entry "new" is expanded and the correct menu entry is found.

The analysis tool marks each step on the respective path whenever the required next action is invoked. Going back on a path, e.g. by closing a menu causes the path to be set back to the respective step on the path. A possible output for the analysis of the theoretical example just mentioned looks like the following:

```
  0% –             [ ] – Toolbar entry
 50% –          [x][ ] – Context menu–menu entry
  0% –          [ ][ ] – File–menu entry
100% –       [x][x][x] – File–new–menu entry
  0% – [ ][ ][ ][ ][ ] – File–new–other–rce–menu entry
 50% –          [x][ ] – Toolbar chevron–menu entry
 75% –    [x][x][x][ ] – Toolbar chevron–other–rce–menu entry
  0% –       [ ][ ][ ] – Toolbar next to chevron–rce–menu entry

  [x] visited step on path        [ ] not visited step on path
```

Listing 3.1: Exemplary output for a Distance from Success Vector

This output can support an analyst to find possible solutions. For instance, path 2 could be improved by reordering the menu entries. Moreover, changing the labels in the general wizard that was opened via path 7 could help to understand the entry's proper meaning.

To this point the focus has been on the conception of the techniques to compare. It is now important to consider the implementation of these concepts.

Chapter 4

Implementation of the Track-Analysis-Tool

"Any fool can write code that a computer can understand.
Good programmers write code that humans can understand."

—Martin Fowler

In this chapter the implementation of the concept of the Track-Analysis-Tool is described. Therefore the technical infrastructure and the data exchange format are introduced. The implementation of the capturing ability to RCE is presented. Eventually the analysis, implementing the subbtask detection and rating algorithms (see 3.2.5—"Analysis Algorithms") is explained.

4.1 Technical Infrastructure

The tool was implemented using the infrastructure established for the development of RCE. To keep track of bugs and features the bug tracker *Mantis* is used. It is web-based and connected to the versioning system *Subversion* [The13].

Additionally the static code analysis tool *Checkstyle* [Che12] is used to enforce common code conventions throughout RCE.

To make sure the code conventions are satisfied in the Subversion repository the tool *Repoguard* [DLR12] is used. Further information can be checked before a commit into the Subversion repository is allowed. Moreover *Jenkins* [Jen13] is used to support continuous integration.

From an organizational point of view two kinds of meetings set the frame. On the one hand weekly meetings of all RCE developers induce the exchange of ideas which turned out to be very productive for the implementation of the tools. Moreover, meetings of smaller teams were

held on demand to focus on specific topics.

The development of the capture functionality and some artifacts required for the user study took place in a separate branch. The branch was extracted from version 2.4.1. Updates caused by the daily business of RCE were not applied during the development to provide comparable results. So the RCE version used in the two iterations of the user study are based on 2.4.1, though 2.5.0 was already available.

Another aspect to be clarified before the actual implementation is the data exchange format between the capture tool and the analysis tool which is described in the following section.

4.2 Data Exchange Format

For the data exchange between the tracking part of RCE and the analysis tool it is desirable to draw on a proven exchange format. This avoids the overhead to design such an exchange format. The chosen data exchange format should be supported by libraries in several programming languages to allow its use on a universal scope. Libraries should be available to make the mapping between the data exchange format and the data structures (e.g. objects) in the respectively programming as seamless as possible. That means changes on one side should not effect the other side.

In order to have as broad functionality as possible it should be possible to use nested data structures in the exchange format, e.g. to represent multiple selections, each having multiple properties. Moreover the format should be legible for human beings. This is particularly important for debugging during the development.

A format that matches the requirements mentioned above is *JSON* [JSO] with *Jackson JSON Processor* [Fas09] as Java library.

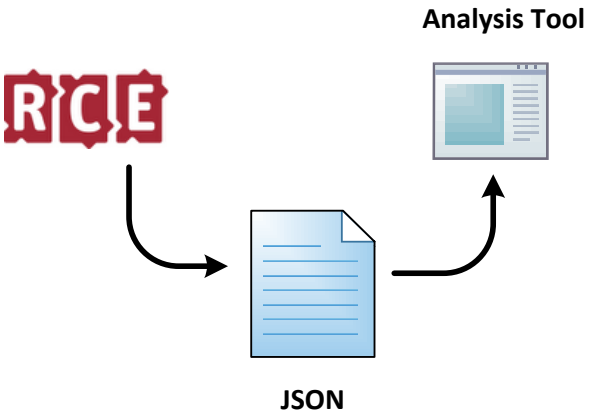

Figure 4.1: JSON as data exchange format between RCE and the Analysis Tool

Figure 4.1 depicts how JSON glues the two parts of the implementation together.

Jackson is a Java library that supports the JSON format and is already used in the context of RCE to handle the processing of configuration files.

To write the interaction events generated during the use of RCE to a log file, the following lines basically suffice, where `jsonGenerator` is an instance keeping the file location and `eventList` is the list that stores the generated interaction events.

```
1 ObjectMapper om = new ObjectMapper();
2 om.writeValue(jsonGenerator, eventList);
```

Listing 4.1: Write into a JSON file

Mapping the interaction events from the log file to Java objects is done within this method:

```
1 private Collection<InteractionEvent>
      readInteractionEventsFromFile(File inputFile){
2   ObjectMapper mapper = new ObjectMapper();
3   Collection<InteractionEvent> events = null;
4      // ...
5   events = mapper.readValue(inputFile, new TypeReference<
        Collection<InteractionEvent>>() {});
6     // ...
7   return events;
8 }
```

Listing 4.2: Read in a JSON file

Thus, the only effort the developer has to expend is to keep the exchanged classes (i.e. `InteractionEvent`) synchronized.

Having examined the infrastructure and the data exchange format it is now described how the capture functionality was added to RCE.

4.3 Capture Tool

The implementation of the capture functionality in RCE is described in this section. Technical challenges are presented and the GUI elements and patterns detectable with this approach are listed. Moreover facts about the mouse track and missing patterns are shown.

4.3.1 Enable Tracking only when needed

To make sure that nobody uses the modified RCE version accidentally and that his or her actions are tracked unintentionally, the tracking has to be activated explicitly.

Therefore a plugin containing a simple dialog was created as an extension to RCE, which can be activated via a menu entry that was defined as an extension point of type *org.eclipse.ui.menus* as shown in Figure 4.2.

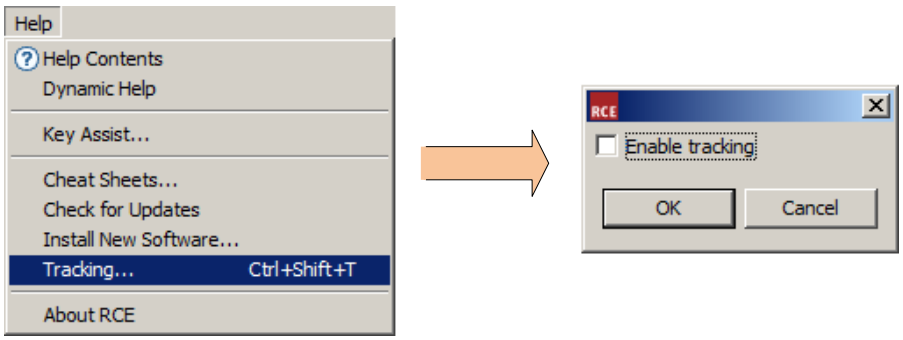

Figure 4.2: Menu entry and dialog added to RCE to prevent accidentally tracking

When the check box in the dialog (see Figure 4.2) is checked and the dialog is confirmed, the tracking begins. In the user study this is the point in time when the task begins.

4.3.2 Implementation-specific Challenges

For SWT and JFace GUI elements it is common to group elements in instances of class `Composite`. `Composites` are controls which are capable of containing other controls. [Ecl11]

Some GUI elements are already instantiated at the start of the application whilst others are created at runtime, e.g. when a menu or a dialog is opened for the first time.

To make sure all GUI elements have their respective listeners registered it is important to first register all listeners for the elements instantiated at application start. Whenever new GUI elements are instantiated at runtime the respective listeners are registered, too.

The general approach of the listener registration is to check each GUI element for its type and then add the according listener, e.g. `SelectionListeners` to buttons or `TreeListeners` to trees. The following recursive function processes the respective GUI elements:

```
1 private void registerListeners(Composite parent){
2   for(Control child : parent.getChildren()){
3     if (child instanceof Composite) {
4       registerListeners(child);
5     } else {
6       checkListenerRegistration(child);
7     }
8   }
9 }
```

Listing 4.3: Java code of recursive registration of listeners

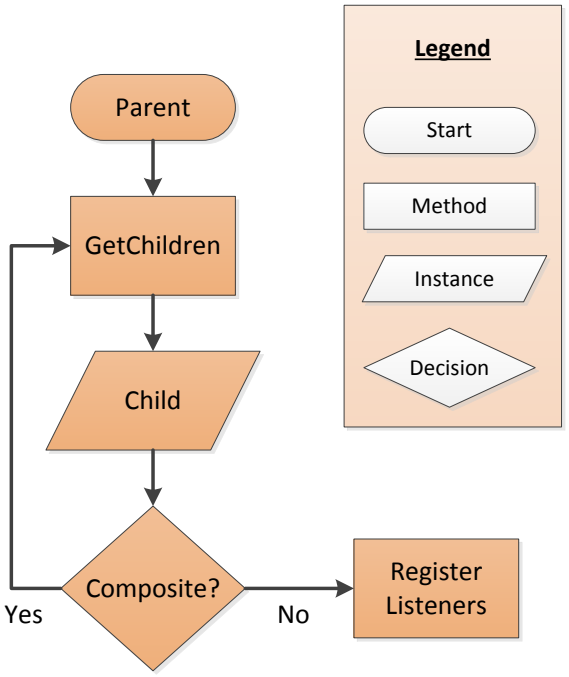

Figure 4.3: Flow chart of the recursive registration of listeners

That means, whenever a Composite is found its contained elements are either processed as Composites again or they are other GUI Elements and it is checked whether a listener has to be registered.

The procedure is depicted as a flow chart in Figure 4.3.

At self-made dialogs it is possible to register the listeners manually. For example, the connection editor is equipped with listeners at several spots to track low level interaction events as well as RCE specific patterns.

Some issues turned out to be not intuitive and are worthwhile being mentioned here:

ArmListeners for Menu Entries

To get a feeling how users interact with the application it is important to identify how and when they browse menus. The event that is triggered when a menu is browsed and entries are highlighted is a so called `ArmEvent`, the respectively called listener and method are `ArmListener` and `widgetArmed(ArmEvent event)`.

As the menu is instantiated when it is opened for the first time, it is not possible to simply add the `ArmListeners` to the menu entries at the start of the application.

Figure 4.4 shows the relations between the listeners and the respective

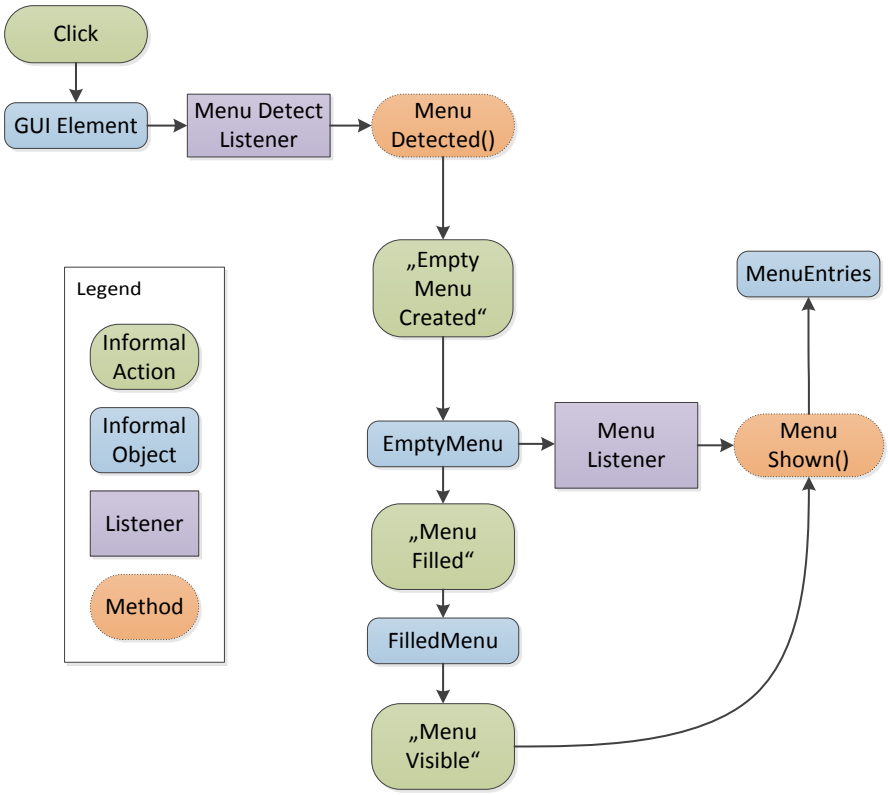

Figure 4.4: Depiction of the registration of listeners for menu entries

methods that are necessary to register `ArmListeners` to menu entries dynamically at runtime. A `MenuDetectListener`'s method `menuDetected()` is invoked whenever a menu is opened via right-click or the toolbar, for instance. Within this method's arguments the menu is contained, but in an empty state. Adding a `MenuListener` to the menu will enable access to the menu entries via the listener's method `menuShown()`. Registering `ArmListeners` to all these menu entries provides the application with the desired functionality to detect browsing through menus.

Native wizards

Wizards are an important and proven concept of RCP. There are many wizards shipped with RCP. These wizards are called "native" wizards henceforth. Tracking the interaction with wizards and dialogs has a vital share to detect and understand usability problems.

For many parts of the application it is possible to hook in listeners programmatically via the pattern shown here:

```
1 if tracking == true then
2 |   registerListeners;
3 else
4 |   //doNothing
5 end
```

Figure 4.5: Pattern for tracking activation

The "tracking" flag is determined in the dialog described in 4.3.1—"Enable Tracking only when needed".

As some parts are either derived from Eclipse built-in classes (like the native wizards) or just added declaratively as extension points it is not possible to apply this pattern. For example, the wizard for new projects is present in RCP applications by default and the original wizard for new workflows consists of standard pages derived from `WizardNewFileCreationPage`.

To hook into these dialogs the workbench is equipped with a `WindowListener`. Whenever a new dialog is opened the method `windowDeactivated()` of this listener is invoked. Inside this method the list of currently existing windows (i.e. instances of `Shell`) in the workbench is processed. With each of the shells as parent element the recursive listener registration method explained in Figure 4.3 is called.

Prevention of Multiple Listener Registration

The relation between GUI elements and listeners is n to m. That means GUI elements can have multiple listeners (also of the same type) and listeners can be added to multiple GUI elements. For the purpose of tracking it is not desirable to have identical listeners added several times to the same GUI elements, as this would result in redundant captured messages.

In order to prevent the registration of multiple identical listeners to a GUI element, each of it is equipped with a flag for each type of listener. This is realized via the `setData()` method that all SWT elements have in common. Before the registration of a listener it is checked whether the flag it set. If the flag is not set already, the listener is registered and the flag is set. If the flag is already set the listener is already registered and nothing needs to be done.

The following code depicts this approach:

```
1 if element.flag == false then
2 |   element.flag = true;
3 |   registerListener(element);
4 else
5 |   //doNothing, Listener already registered
6 end
```

Figure 4.6: Prevention of multiple listener registration

4.3.3 SWT Components and Events

The following list gives an overview about the covered GUI elements from the SWT framework and their respective events.

SWT Elements	Captured Events
CTabFolder	Selection event
Specific Text field	Key event
Button	Selection event
Any Control	Mouse event
Any Control	Focus event
Any Control	Key event
Combo	Selection event
Tree	Open event, Tree event, Drag event
Any Control	Traverse event
Menu item	Arm event, Selection event

Table 4.1: SWT elements and events captured from them

This list is small compared to the more exhaustive list presented in 3.2.3—"Interaction Patterns" and Appendix B—"List of Typical GUI Elements and Interaction Patterns". Nevertheless, it turned out to be sufficient to detect many usability problems.

4.3.4 GEF Components and Events

GEF Element	Captured Events
PaletteViewer	Active tool changed, Mouse event
GraphicalViewer	Selection changed event
EditPart	Removing child, Adding child

Table 4.2: GEF elements and events captured from them

The *Graphical Editing Framework* (GEF) is used for the workflow editor. Thus, interaction events dealing with the selection of tools (e.g. workflow components) in the palette or with the interaction with the workflow components on the canvas (adding, removing, selecting) are considered.

4.3.5 RCE-specific Patterns

In addition to the low level events like mouse clicks or selections events, so called patterns are tracked on a higher level of abstraction. Patterns in the context of RCE can be the following:

RCE Element	Action
Project	created
New Workflow Wizard	started
Workflow	created, executed
Workflow Editor	opened
Workflow Component	added, deleted
Endpoint	added, edited, deleted
Connection	added, deleted

Table 4.3: Captured patterns

These patterns are intended to give a broader overview about the problems the probands might face. Comparisons between the points in time at which the patterns are invoked might help to understand problems on a broader scale, similar to the times for the subtasks.

4.3.6 Mouse Track Events

In order to create visualizations of the mouse track it is necessary to log mouse positions. Therefore instances of the class `MouseSpot` are created and added to a list that is written into a separate file in JSON format in the same way as the interaction events. The class `MouseSpot` is a plain data class holding the properties:

- Time stamp
- X-Coordinate
- Y-Coordinate
- Reporting class

X and Y coordinates are important for the visualization. The time stamp is necessary to assign mouse spots to subtasks. In the analysis all mouse spots with a time stamp in the period of a subtask are assigned to this subtask. The reporting class is necessary for debugging purposes the make sure that all windows and dialogs contribute to the tracking of mouse spots.

It is important to make sure that each window is equipped with `MouseMovementListeners`. They are added in the same manner as the other listeners. The mouse events the mouse movement listeners receive contain coordinates relative to the window's position. Thus, the position of the window must be added as offset.

4.3.7 Missing Events

For the prototypical implementation of the tracking tool several interaction events that might be interesting for a deeper understanding but were not crucial to the analysis were not considered.

One interaction event that turned out to be invoked by the probands in the user study very frequently was the browsing of the tool palette of the graphical editor. While the browsing of the menus and toolbars is covered by means of SWT `ArmListeners` (see Figure 4.4) the palette is part of the GEF framework and does not provide this functionality. As alternative, a mouse listener with the according `mouseHover()` method was registered. The respective event is invoked when the mouse stays on one palette entry for more than a second which turned out to be a too long time span.

Another missing interaction pattern in the context of GEF is drag and drop within the graphical editor. Tracking the movements of workflow components over the canvas might allow to draw further conclusions.

Moreover views that do not contribute to the user task are not included in the tracking process. For example, the workflow browser, an RCE specific view displaying information about past workflow executions, is not included explicitly. Nevertheless, when probands spend a lot of time in this view the elapsing time is tracked but events invoked within this view are not tracked as they are assumed to provide little information.

Text fields that contribute to the user task are equipped with a property holding a representative name. Text fields missing that property can not be determined exactly but are generally assumed to be "unproductive" text fields in the context of the user task. When text is entered into such an "unproductive" text field this is regarded as a negative event.

Considering the mouse track it is not possible to track mouse movements faster than the interval of the driver. That means when a proband moves the mouse very fast over the screen the track becomes porous.

Though all of these points are not assumed to reveal crucial usability problems it would be a nice addition to the set of tracked events to create a fundamental understanding about the users' interactions.

In the above presentation the realization of the capture ability in RCE was shown. Now it is important to analyze this captured interaction data. Therefore the analysis tool is presented hereinafter.

4.4 Analysis Tool

The analysis tool implementing the algorithms described in 3.2.5—"Analysis Algorithms" was written in Java. When the analysis tool

starts a dialog for file selection is opened. Therefore the predefined SWT class `FileDialog` is used. When the user picks a file that contains interaction data the corresponding file for the mouse track is determined by using the time stamps that initiates the file name. A filter showing only files with the extension "json" by default is applied (see Figure 4.7).

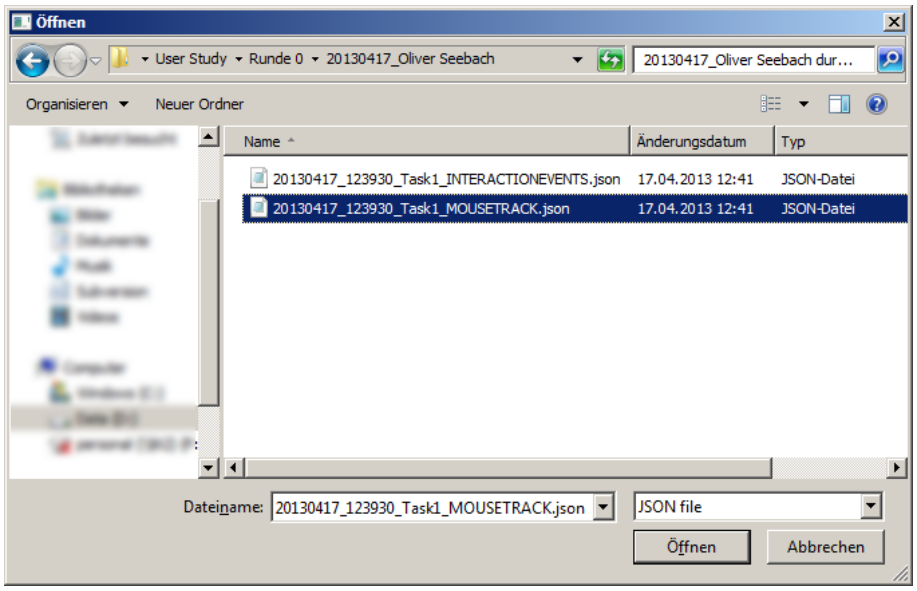

Figure 4.7: Dialog to choose a file to be analyzed

The following component diagram depicts the way the analysis tool:

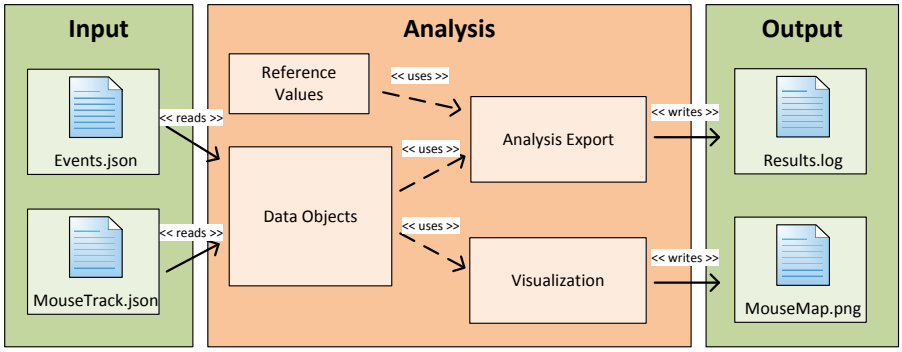

Figure 4.8: Component diagram of the Analysis Tool

The structure of the analysis follows the Input-Process-Output (IPO) Model.

The input is the captured interaction events and the captured mouse location stored in JSON-formatted files. The output is an analysis report as a text file and the mouse track visualizations as image files. In the analysis the events and mouse locations are mapped to data objects using the Jackson library.

The data objects are `InteractionEvent` and `MouseSpot`. The analysis uses

a data object `Subtask` to group them. The analysis uses reference values to enhance the report. In the current implementation they are defined programmatically, but adding them via configuration files is also imaginable. The analysis results are written to the file system as a report in text format and the visualizations are stored in the same folder.

4.4.1 Subtask Detection

The events are sequentially assigned to the currently detected subtask using the algorithm in 3.2.5—"Algorithm for Subtask Detection". Thereby for each event it is checked whether the event can potentially cause the beginning of a subtask and if all conditions to fulfill the preceding subtask are met. When these two conditions are fulfilled the event currently under observation is added to the current subtask, then the subtask identifier is switched. This is necessary to assign the event that fulfills a subtask to this subtask and not to the next one.

4.4.2 Subtask Rating

After this assignment the list of events per subtask is rated depending on the rating list for each subtask (see 3.3—"Algorithm for Subtask Rating"). For the prototypical implementation in the course of the thesis it was decided to add the rating lists per subtask programmatically. I.e. for each event several conditionals are checked whether the event should be rated positive, neutral or negative. For this single task this approach suffices, but as a general approach it would be desirable to configure the conditions for the respective rating in a centralized configuration file in a human readable manner.

Additionally, the idle time for the entire task was calculated. Therefore the time stamps of the tracked mouse spots and interaction events where combined and chronologically ordered. If the time span between two time stamps was longer than a certain threshold this time span was regarded as idle time and added to the idle time counter.

This idle time counter would also trigger if the proband is overwhelmed by the application and keeps staring on the screen. But the user study showed idle time is most times triggered when the user does not work on the task, e.g. when turning around to the supervisor and giving feedback.

4.4.3 Result Presentation

Within the prototypical implementation it was decided to print the results on the console and to place a copy of the printed messages in a text file in the workspace of the project in the Eclipse IDE. An excerpt from a prototypical output of the analysis can be found in Appendix C—"Excerpt from Example Results Of the Analysis". To beautify the output and to enhance the readability for a usability expert using the analysis tool it would be desirable to add output in a nicely formatted

PDF or HTML file. The textual output consists of several parts:

- Log date: The time stamp when the log started
- Analysis date: The time stamp when the analysis was executed
- Overall amount of subtasks and events, overall duration and idle time
- Short description of the subtasks
- Short overview over each subtask, consisting of:
  - The amount of events
  - The duration
  - An optimal duration reference value
  - The amount of positive, neutral and negative events
  - The confusion warning flag
  - The amount of distinct events
  - The amount per event type
- Sequence of events per subtask, containing an ID, the rating, the event type and (if available) further information, e.g. the typed key or the label of the selected button.
- Sequence of patterns, time passed since the last pattern, a short human readable description of the pattern
- Screen resolution: The resolution of the proband's primary monitor
- Save location: The location where the created screenshots are saved
- Overall mouse track length in pixel
- List of mouse spots per subtask

With this output at hand it is possible to identify many usability problems, that are not visible from the raw captured data. The context added to the data is one reason for that. Events that are positive in some context might be negative or neutral in another context and vice versa.

Moreover events that are assumed to be not crucial in this context (`FocusEvents`, `MouseEvents` and `ArmEvents` are filtered and only included in the calculation of a possible confusion warning. The following procedure to browse the output turned out to be effective:

First, in the overview subtasks with both negative events and confusion warning are sought. Then having a closer look at the chronologically ordered events is assumed to allow more understanding about possible usability problems.

The same procedure is repeated for subtasks only having either negatively rated events or confusion warnings and then for subtasks that took significantly longer than the given optimal value. Comparing the actual times to the optimal times is rather error prone and can often

be skipped because these times vary a lot and might be influenced by many other factors (general working speed, reading performance, current concentration).

Currently the rating of events has the three states positive, neutral or negative. More diverse ratings would allow finer nuances and a more precise overview. For example, it would be possible to rate pressing backspace (to delete text field content) with a -0.5 as it is possibly just a sign for a typo (but can also be a sign for misunderstanding the meaning of a text field, so 0 would be too optimistic). Furthermore events that are clearly a sign for a usability problem (canceling a wizard, for instance) can be accredited with a rating of -2.

But as these more granular ratings would be set arbitrary without more exhaustive studies which would exceed the scope of the thesis they are not implemented.

4.4.4 Mouse Track

Another means to provide further inside about possible usability problems is a visualization of the mouse track.

There are several visualization techniques imaginable, for example several variations of heat maps, plots with lines connecting the single spots or plots showing the all single saved spots as circles or crosses. For sake of simplicity it was decided to plot the single spots as small dots. Figure 4.9 shows an example of a plotted mouse track of a single subtask.

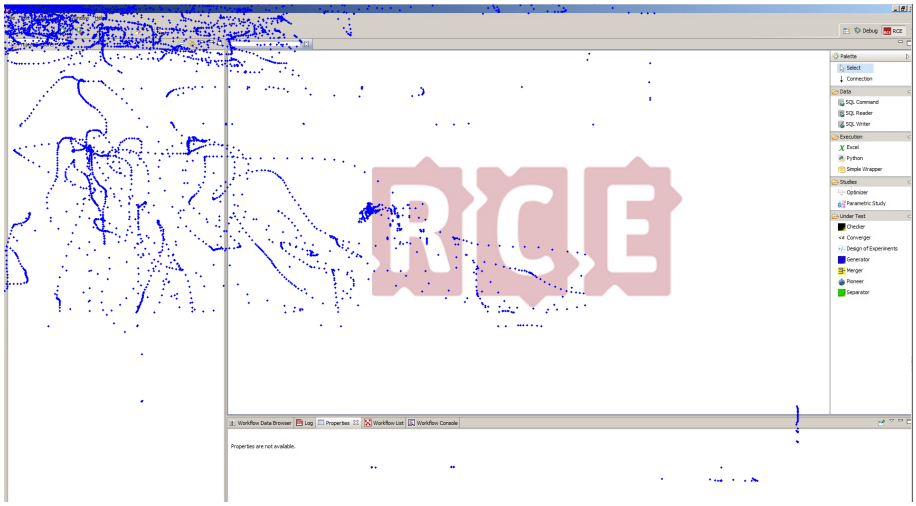

Figure 4.9: Exemplary mouse track visualization

As background a screenshot of the RCE surface with standard dimensions and standard arrangement was added in the respective screen resolution of the proband. It was assumed that the views are not moved and that the window is maximized. The user study showed that these assumptions are appropriate.

The proband's resolution is determined by an interaction event of type `ControlEvent` which is invoked when the tracking starts. It was assumed that the probands work on their primary monitor so this resolution was taken into account here. In study it turned out that some probands worked on their secondary monitor so the proper resolution of the application had to be determined manually.

Technically the plot was realized with SWT. On an instance of the type `Canvas` with the means of the *GraphicalContext* (GC) of *SWT Graphics* dots are drawn at the locations of the tracked mouse spots.

This is done for each subtask. If the subtask contains the usage of a wizard or a dialog the background image shows the application with the respective wizard or dialog open. The resulting canvas is printed to a file using the capability of the class `Robot` of the *Abstract Window Toolkit* (AWT) to take screenshots.

The implemented tracking functionality and the analysis tool enable an automated approach to analyze usability problems. This is applied in the user study which is presented in the next chapter.

Chapter 5

Usability Study

"I long to accomplish a great and noble task, but it is my chief duty to accomplish small tasks as if they were great and noble."

—Helen Keller

In order to evaluate to which extent the Track-Analysis-Tool developed in the course of the thesis can compete with the proven Think Aloud method, a user study applying both techniques simultaneously is conducted.

The study is conducted as an exploratory (sometimes called formative) study [RCS11, p. 29 ff][TA10, p. 45 ff]. Exploratory studies are particularly useful when several designs are to be examined and a finalization of the product takes a while. This chapter contains the setup, the probands and the task of the user study. Questionnaires and the methodology to create design improvement are described.

5.1 Application of the Track-Analysis-Tool and the Think Aloud Method

In 3.3.2—"Comparison of Existing Techniques" it is reasoned why the Think Aloud approach is a suitable reference usability method here. To recall, the main reasons for that are listed here:

As neither the author or anyone in RCE development team is a usability expert, all reference techniques requiring a high level of usability expertise are discarded as candidates.

Another requirement is a small number of required probands. In combination with the small number of evaluators required, the Think Aloud method became the reference technique of choice.

There are arguments that the Think Aloud approach can decrease

the user's performance because of the overhead of verbalizing the thoughts, but others argue that the verbalization stimulates a very conscious way of working and thus improves performance.

To counterbalance the drawbacks the fact of only one supervisor brings in, the execution of the user task is videotaped. Then the analysis can be done afterwards, notes do not have to be taken while the user executes the task and it becomes less likely that information is missed.

5.1.1 Distinction between User and Proband

In order to prevent confusion about the terms *User* and *Proband* the distinction as it is made in the thesis is clarified here:

A *User* refers to a general user of the product independent from any studies. In contrast to that a person taking part in a study is regarded as a *Proband*.

5.2 Setup

In this section the setup of the study is described. This contains preparations and information about the probands.

5.2.1 Infrastructure

The user study was conducted at the proband's workplace. If a proband works on a different workplace, several circumstances might have an impact: An unfamiliar chair, a different keyboard layout, a different screen contrast and resolution, a different mouse or other lighting conditions. In order to prevent such bias, the technical infrastructure was set up at every proband's workplace.

A camera was placed behind the proband, so that the screen is videotaped. The proband's voice and thus the verbalized thoughts are captured. Additional videotaping of the proband's facial expressions was omitted as the emotional reactions are assumed to be covered by the verbalization.

Figure 5.1 sketches the arrangement of a typical workplace.

In order to minimize distractions by other people in the room, there is just one supervisor present. The proband is requested to fill in the questionnaire about her or his background knowledge.

Then a version of RCE containing the tracking plug-in is installed on the proband's machine, started and maximized to have comparable conditions. Note that in the second round the installed RCE version contains the design improvements derived from the first round. Thereafter, the proband is requested to read the introduction to RCE and the user study, to get some feeling about the context (see Appendix G—

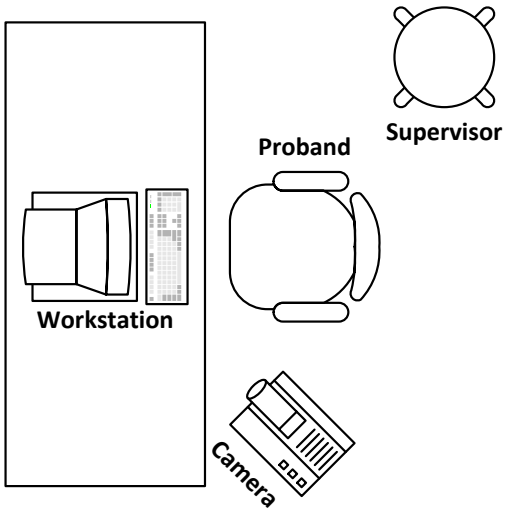

Figure 5.1: Sketch of the setup for the user test

"General Information About User Task". The proband activates the tracking, starts to read the user task and processes the items on the task step by step (see Appendix D—"User Task Sheet"). When the proband works on the task and gets stuck so that she or he would not be able to continue with the task, a hint is given.

After the task is finished the questionnaire inquiring the satisfaction is filled out and the log file is copied to a storage device (see Appendix E—"Questionnaire about Satisfaction").

5.2.2 Users

Potential RCE users were chosen as probands. For sake of simplicity only employees of DLR were chosen. There were no specific requirements regarding prior knowledge except for one: The probands were supposed to have as little knowledge as possible about working with RCE. This assumption was confirmed by the results of the questionnaire regarding previous knowledge (see Appendix F—"Questionnaire about Foreknowledge").

The questionnaire yielded that all probands felt rather experienced in working with computers. The probands' average age was 33.6 years, and youngest proband was 24 and the oldest proband was 50. Regarding the profession computer scientists and business economists were predominant. Eight of the probands were male while 3 were female. The probands are from two different departments in the DLR, working on different domains. The native language of all probands is German.

As a reference, in each round one proband familiar with RCE was tested. Their metrics are not included in the evaluation. Probands were only employed to one round, as otherwise learning effects would have

biased the results.

After the setup is clarified, the task the probands are required to solve is explained.

5.3 User Task

For the study the users had to fulfill a single task. This has the advantage that it is possible to conduct a test with more probands within the temporal restrictions of the thesis. Moreover effects of exhaustion do not have to be taken into account. It is assumed that typical usability problems will occur also in a single task.

The task should cover many typical interactions with RCE. It does not aim to detect problems related to specific workflow components, but to discover problems occurring with the RCE framework itself. Note that in the context of RCE and the user study "workflow components" refer to a tool that can be added to a workflow. It must not be confused with software components as presented in 4.4—"Analysis Tool".

Typical interactions with RCE are the following:

Interaction
Create, copy, paste, delete, rename a project folder
Create, copy, paste, delete, rename a workflow file
Create, copy, paste, delete, rename workflow data
Open a workflow in the workflow editor
Add, copy, delete, move, rename a workflow component
Configure inputs and outputs of a workflow component
Configure workflow component specific options
Create channels between workflow components
Execute a workflow
Check outcome of a workflow

Table 5.1: List of typical interactions with RCE

The task to be defined is a tradeoff between complexity and closeness to reality. On the one hand, if the task is too complex users might get stuck because the task itself is not suitable to be done within appropriate time in a user study or without a briefing. On the other hand users might not take their job seriously if the task is trivial or meaningless.

In the given task three workflow components created for the user study have to be used. The generator component generates integer values in a range defined by the user. The separator component takes a stream of values as input and provides only the even numbers as output for the next workflow component. The checker component verifies whether the received stream equals the desired sequence of numbers. If the checker component succeeds a dialog informs the user about the success.

Actions the user has to accomplish are: Creating a project, creating a workflow, open a workflow in the workflow editor, configure inputs and outputs of a workflow component, configure workflow component specific options via text fields and check boxes, create channels between workflow components, execute a workflow.

Other interactions are also possible, trackable and might lead to success, but this is the minimal set of interactions.

The sheet as is was handed out the users is attached in Appendix G—"General Information About User Task". To avoid problems that the users might have with the wording in English, the sheets are in German.

The instructions listed on the sheet differ from the list of subtasks introduced in 3.2.2—"Subtask Segmentation". The reason is that there are different scopes related with the lists. While the list of instructions is supposed to simulate typical goals from a user's point of view, the list of subtasks has a more technical focus. For example, to fulfill the first instruction "Create a workflow" four subtasks have to be fulfilled. Stating these four subtasks explicitly would be a possibility, but it would bias the results. When the user wants to create a workflow his or her mental model does possibly not include the idea of creating a project first.

To determine the previous knowledge of the probands and the satisfaction when working on the user task presented here two questionnaires have been defined. They are presented in the next section.

5.4 Questionnaires

The users were asked to fill out two questionnaires. To prevent misunderstandings caused by the language they are written in German.

The first one dealing with background information about the users was presented before working with the application, to prevent distraction. It can be found in Appendix F—"Questionnaire about Foreknowledge". Information about technical prior knowledge was acquired. By doing so reasons for positive or negative outliers could be found.

For example, it could be the case that people familiar with the Eclipse IDE know that in most cases projects are required as root node before any application specific files can be created. Moreover, performance tendencies regarding age and general computer knowledge could be of interest.

The second questionnaire measures satisfaction as a metric. It was presented after the probands worked with RCE so their feeling when working with it was still present. It is desirable that the satisfaction metrics improve per iteration. The questionnaire contains seven ques-

tions adapted from the Computer Usability Satisfaction Questionnaires as described in 3.4.2—"Standardized Metrics"

To determine whether the satisfaction measured with the questionnaires improve an iterative approach is useful. This is explained in the following section.

5.5 Iterative Approach

The user study is planned as an iterative procedure. The Think Aloud method applied here is suited for this approach as it provides sufficient insight into the nature of the problems to suggest specific changes to the interface. [Nie94b, p. 105]

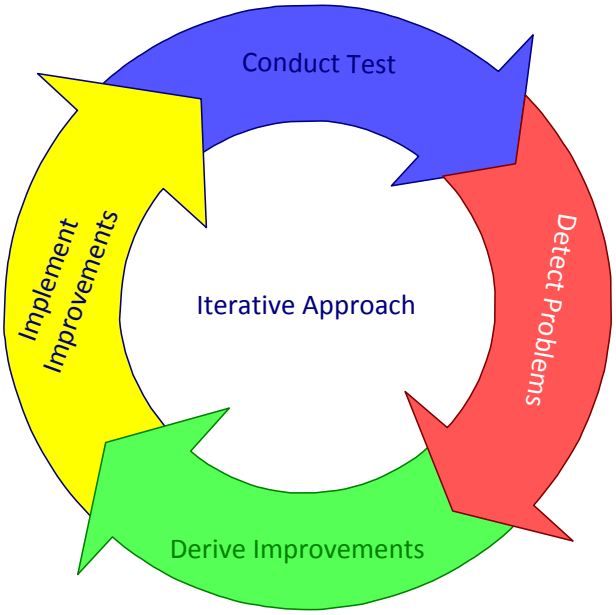

Figure 5.2: Depiction of the iterative approach and its phases

Figure 5.2 depicts the steps per iteration derived from the approach presented in [Per07]: Conduct the test, detect usability problems, derive design improvements, implement design improvements.

In theory, the iteration can be repeated either until no more usability problems are detected or until the metrics do not improve anymore.

Due to the temporal restrictions of the thesis two iterations are executed. Though revealing less exhaustive results it is assumed that tendencies are recognizable.

The problem detection procedure mentioned in this section is presented hereinafter.

5.6 Problem Detection Procedure

During the usability study several usability problems were detected. The usability problems had different degrees of severity and different frequencies. The severity was assigned by the author based on the impact the detected problem had on the task accomplishment during the usability study. The severity ranges from 1 to 10 where 10 is a blocking problem that could only be resolved by a hint and 1 is something like a typo or labeling issue that a proband stumbled over.

Note that there are other notions of severity. For example, [Nie94b, p. 102 ff] understands severity as composed of frequency and impact. For sake of simplicity the definition of severity mentioned above is referred to henceforth.

In order to find the problems that were crucial to the task the frequent (i.e. they occured at least twice) and the severe (degree of severity of at least 5) problems were examined in more detail and design improvements were suggested.

5.7 Design Improvements

It is assumed that good design improvements are better found in a group discussion than by a single person. Based on the list of severe or frequent problems possible solutions were discussed in the group of RCE developers. Thereby the creativity method *Six Thinking Hats* (see 2.5—"Creativity Method Six Thinking Hats") was applied.

The entire group is required to have a look at a problem with a certain attitude, ignoring the other attitudes. The six hats represent six different attitudes, which are: Organize the thinking, neutral or objective thinking, creative thinking, positive thinking, negative thinking, emotional thinking.

It turned out that some of the hats were experienced as artificial or too similar to each other in this context. Other approaches, like the *Walt Disney* method [SB12, p. 276] are more minimalistic and are probably perceived as more natural in this context.

A particularly useful aspect is the green and yellow hat which stand for creativity and optimism. Hereby many valuable ideas are created as any technological, temporal or monetary boundaries are ignored.

The composition of members of the development team turned out to fit good here. There are members that have years of experience with RCE but might be stuck with some control concepts. Other new members of the RCE development team lack the experience with the application but bring in new creative ideas as they have an impartial view on the issues.

Ideas for design improvements were analyzed and a list with crucial and optional design improvements was created. Due to the restrictions of the thesis' schedule some improvements could not be realized.

5.8 Open Design Improvements

For some problems no design improvements have been implemented. This has several reasons. First, the temporal restrictions limit the improvements and second, sometimes no global solution is apparent. The solutions implemented here have the requirement to be applicable on the global scope and not for the specific use case of the user study.

A usability problem that has not been tackled is the missing feedback when values are entered to configure a workflow component. There are some solutions that appear suitable at first sight, but have drawbacks on the global scope. Thus, no changes are implemented and the usability problem is expected to remain.

Having examined the setup and procedure of the user study it now necessary to discuss the results.

Chapter 6

Discussion of the Results

"However beautiful the strategy, you should occasionally look at the results."

—Winston Churchill

In this chapter the results of the thesis are discussed. Thereby the capabilities of the log analysis and the think aloud technique are compared. Moreover the metrics before and after the design improvements are compared. Limitations of the results and possible bias are explained.

6.1 Identified Usability Problems in the First Round

All problems detected in the first round occurred at least one and at most five times. There were 30 distinct problems detected with an unique count of 53 usability problems.

11 frequent and 12 severe problems were detected. As some problems were severe and frequent a total of 15 severe or frequent problems were under observation. To enhance the solution generation the problems were grouped into categories so that possible problems on a different level of abstraction might become obvious.

Five categories of problems were formed:

- Creation of projects and workflows
- Recognition that the workflow editor is open
- Add workflow components to the workflow
- Workflow component configuration
- Connect workflow components

For a more exhaustive list containing detailed identified problems, see Appendix H—"List of Frequent and Severe Usability Problems".

Derived from these concrete problems concrete design improvements have been realized, as described hereinafter.

6.2 Realized Design Improvements

The concrete realized design improvements are the following:

Creation of Projects and Workflows

The most crucial problem the probands faced was that the concept of projects and other files was not intuitive. Only one of the six probands was familiar with the concept that you need a project as parent instance for other files in Eclipse RCP based applications.

The task intentionally says "Create a workflow" and not "Create a project and then create a workflow" because this is assumed to match the user's plan more accurate: The user just wants to create a workflow. The new designed workflow wizard intends to assist the user in creating the workflow by "placing" a workflow inside a project. This project can either be an existing one or it is implicitly created.

Once the structure of a project containing a workflow is created, it is assumed to be easier to learn for the users. Figure 6.1 shows a screenshot of the new designed workflow wizard:

Recognition that the Workflow Editor is open

When a new empty workflow editor is opened an additional label on the canvas of the workflow editor was added that gives a hint that workflow components can be added on the canvas. The label disappears when the first workflow component is added to the workflow.

Add Workflow Components to the Workflow

As some users tried to add workflow components to the canvas via double-click as a common pattern used in other applications this functionality was added. A double-click adds the selected workflow component to the upper left corner of the canvas, if it intersects with an already present workflow component it is shifted by an offset until a free spot is detected.

Moreover the label to recognize the workflow editor mentioned above is assumed to help to understand the way workflow components are added.

Find and Execute the Workflow Component Configuration

A fact that made it hard for the users to find the configuration of a workflow component is that there is only one way to open the tab where the configuration is located. A double-click on the workflow

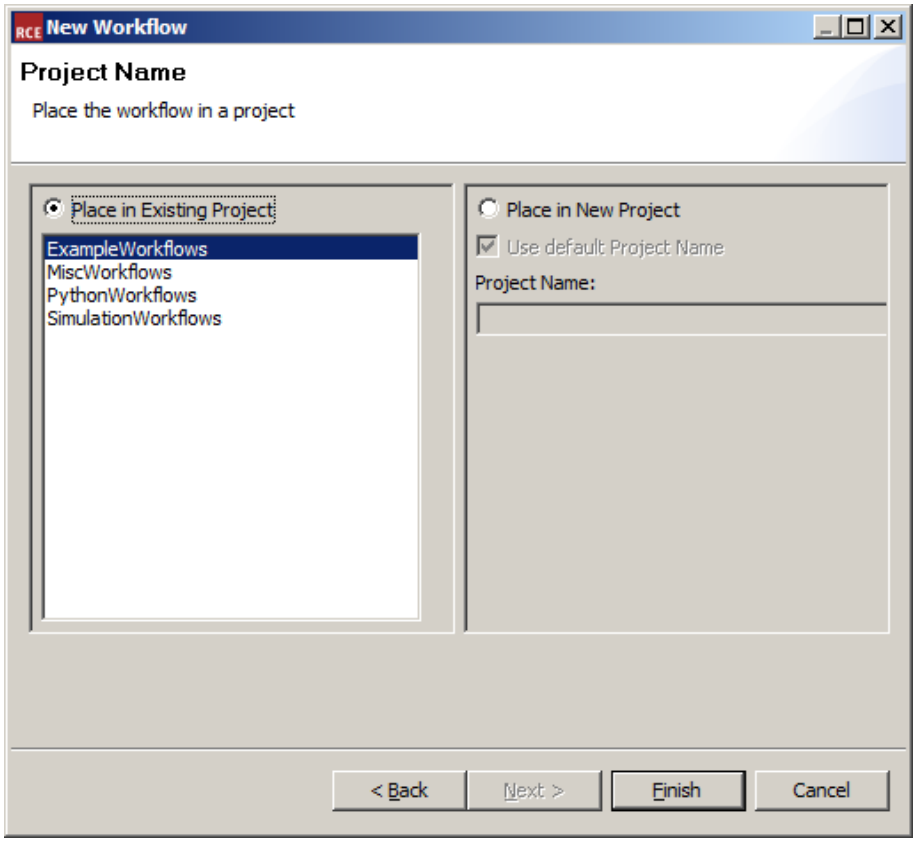

Figure 6.1: Screenshot of the redesigned workflow wizard: Workflows can be placed inside a project

component is the only possibility to open the configuration. As most probands first searched for the configuration in the context menu of the workflow component, a menu entry "Open Configuration" was added.

Moreover, the context menu was reorganized and the RCE specific entries were added to a group delimited by separators (see Figure 6.2).

Connect Workflow Components

To pull connections a dialog called *Connection Editor* is used. In the improved design the connection editor was made accessible not only from the context menu of the workflow editor but also via an entry in the palette.

Once the connection editor is opened in the original design drag and drop was the only possible way to connect the workflow components. As the insertion of workflow components to the workflow editor is possible via drag and drop as well as via point and click this behavior should be consistent. Workflow components can be connected via drag and drop in the improved design.

Another issue appeared when probands entered the connection editor

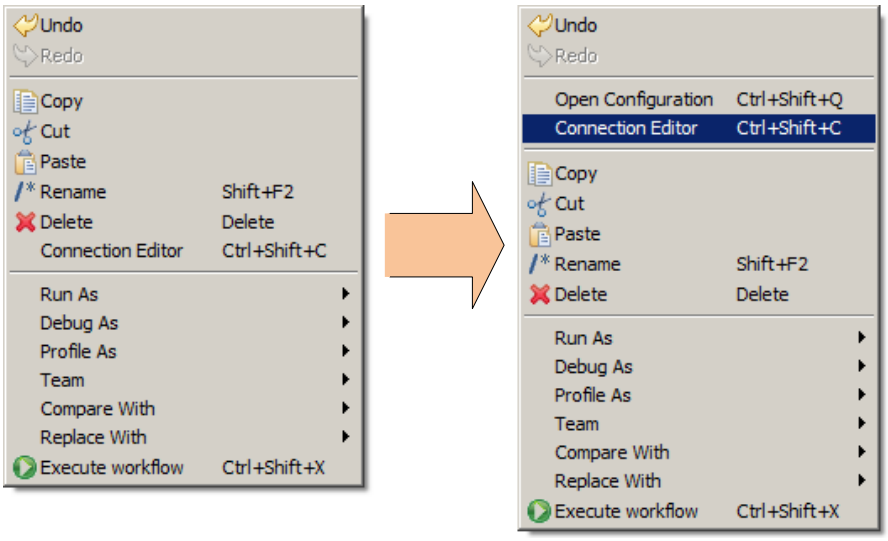

Figure 6.2: Screenshot of the redesigned component's context menu: the configuration can be opened via a menu entry

via another entry point. There is a palette entry "Connection" that opens - when selecting two workflow components - a connection editor reduced to the two components about to be connected. For the probands it was not clear why they had to pull connections twice if their choice is unique. So in the improved design the connections are automatically shown in the connection editor if there is only one output in the source component and only one input in the target component and the types match.

The palette entry "Connection" was renamed into "Pull Connection", an entry to open the connection editor was added (see Figure 6.3).

The identified problems and redesign suggestions give an insight in the capabilities of the approaches used to detect them. An detailed evaluation of the findings is conducted in the following section.

6.3 Evaluation

This section contains the evaluation of the results of the two iterations conducted within the user study. The evaluation focuses on three aspects:

- What are the commonalities and the differences of the problems detected by the Track-Analysis-Tool and the Think Aloud approach?
- How did the number of identified problems change between the iterations?
- Did the metrics improve between the iterations?

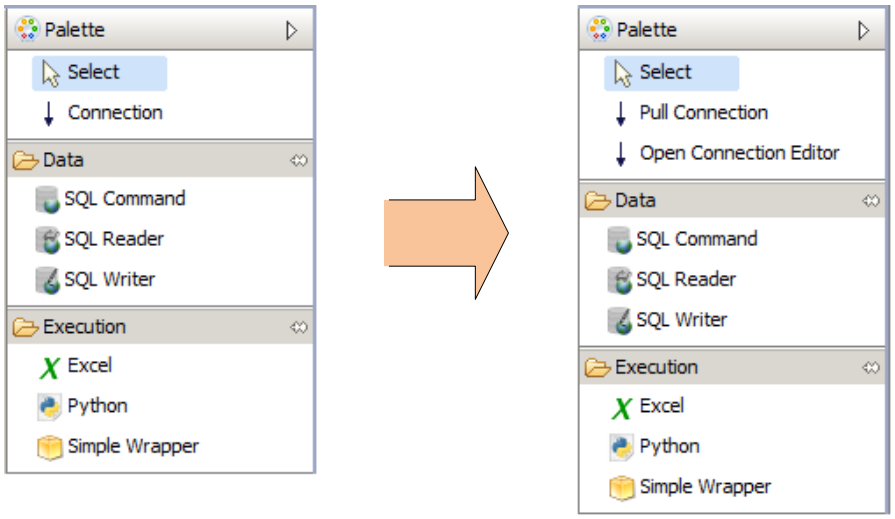

Figure 6.3: Screenshot of the redesigned palette: The connection editor can be directly accessed

The evaluation aims to clarify in which situations an automated analysis tool such as the Track-Analysis-Tool is suitable and when a more labor-intensive approach like a Think Aloud method should applied.

6.3.1 Comparison between Track-Analysis-Tool and Think Aloud Method

The examination of the results of the Track-Analysis-Tool and the Think Aloud approach yielded several commonalities and several differences. Some of are likely to be accountable to the setup, others appear to be inherent to the two approaches and yet others are caused by the fact that the implementation is prototypical and that some functionalities have flaws.

General Differences

In general one can say that the Track-Analysis-Tool reveals less usability problems than the Think Aloud approach.

The problems found by the Track-Analysis-Tool are on a higher level of abstraction. The most crucial usability problem of the first iteration serves as an example here. The probands had problems with the principle that for the creation of any folder or file a project as containing structure has to be added. I.e. in order to proceed first a project and then a workflow has to be created. This does not match the intended actions of the probands as they only want to create one instance, namely the workflow. So it happened that probands created a project and assumed that this was already the workflow. On the video this is clearly visible and audible as people commented the action of creating a project like "And now I think I created the workflow". On the results of the

Track-Analysis-Tool this information is also visible as one can see that the text field containing the project's name is filled like *"Workflow"* (see 6.1—"Excerpt from the analysis results: Project naming").

```
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - W
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - o
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - r
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - k
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - f
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - l
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - o
KeyEvent - keyPressed - Text {} - ProjectNameTextfield - w
```

Listing 6.1: Excerpt from the analysis results: Project naming

Additionally one can observe from the large number of arm events in the analyzed log that the user was not able to go on with the task and thus started seeking (see 6.2—"Excerpt from the analysis results: Seeking funtionality").

```
ArmEvent - MenuItem {&Help} - Help
ArmEvent - MenuItem {&Help Contents} - Help Contents
ArmEvent - MenuItem {&Help} - Help
ArmEvent - MenuItem {&Window} - Window
ArmEvent - MenuItem {&Run} - Run
ArmEvent - MenuItem {Se&arch} - Search
ArmEvent - MenuItem {&Edit} - Edit
ArmEvent - MenuItem {&File} - File
ArmEvent - MenuItem {New} - New
```

Listing 6.2: Excerpt from the analysis results: Seeking funtionality

This can lead to the conclusion that all actions vital to the creation of a workflow should undergo an inspection and the affected GUI elements should be redesigned.

Given that is was already speculated that the steps to create a workflow might cause trouble, a first use case for the Track-Analysis-Tool can be derived: If there is a suspicion that a part of the application is problematic, the Track-Analysis-Tool can substantiate this suspicion.

In this context the first difference between the two approaches becomes visible. While the Track-Analysis-Tool just reveals that there is a problem with the creation of workflows, the Think Aloud study is able to gather more distinct problems. Several problems on a lower level of abstraction are found.

For example, probands had problems mixing up the wizards for a new project and for a new workflow, or probands did not find where they can add a workflow to a project or they did not even know that adding the workflow to the project is required as next step. These problems were concluded to a more general problem.

Recalling that the Track-Analysis-Tool yields the problems on that

higher level of abstraction it can be seen as another benefit of the Track-Analysis-Tool over the Think Aloud approach: While the Think Aloud approach tends to find problems on a low level of abstraction problems on a more global scale can be missed. But on the other hand it can happen that the Track-Analysis-Tool yields problems on a too high level of abstraction, that makes concrete design improvements problematic or that does not reveal any further information. The knowledge about the general presence of a usability problem in the application under observation is one of the requirements for the approach.

As another difference of the two approaches the intended actions of the users are differently apparent. Understanding the intended actions can be vital to find clues where to put additional menu entries or additional functionality the user expected. While browsing of menus can be understood by examining the arm event from the Track-Analysis-Tool as well as from the video, in other situations the Track-Analysis-Tool has drawbacks.

As a good example, the setting of the generated numbers for the generator component is mentioned here. After the creation of the output channel the probands are requested to enter the first and last number to be generated. This is done in a separate tab in the properties view. Some probands tried to define these borders by editing the output they created beforehand. From the captured data it can not be determined whether this was the actual intention or whether concepts are mixed up. On the video it is clearly recognizable that the proband experiences confusion here.

Derived from the experiences made during the examination of the analyzed log and the videos another difference between the approaches becomes clear. For efficient analysis of the output of the Track-Analysis-Tool fundamental knowledge about the context is required. For example, in the chronological listing of events per subtask, an arm event is represented by the event type (i.e. `ArmEvent`) and the label of the hovered menu item. If the analyst is familiar with the context this suffices and the analyzed log can be examined efficiently. But if the analyst is not familiar with the context and has to seek every menu entry in the application this can make an examination tedious or even impossible.

Moreover, the interpretation of the results of the Track-Analysis-Tool is often ambiguous. The probands often missed feedback whether their entered values are saved or not. In search of confirmation, they browsed the other tabs in the properties view, particularly the tab labeled "advanced". In the video the reasons for that are clear as probands verbalized their problems. Though the opening of the advanced tab is regarded as a negative event, because it is not necessary to solve the task, it could also be opened out of curiosity. Similarly, the issue mentioned in the paragraph above, is an example for the interpretation of an event sequence.

Appending a degree of severity to each of the identified usability problems helps to prioritize the redesign implementations. Using the Track-Analysis-Tool it is not possible to determine the severity of the problems. While in the video one can see how much impact the problem had and how much the proband had to struggle this is missing in the captured data. The missing reactions on an emotional level (insecurity, anger, joy) are a reason for that.

When it comes to the quantification of values, the Track-Analysis-Tool is superior to the Think Aloud method. Concrete durations or amounts of certain events can be gathered in an automated fashion. They can be determined by analyzing the video as well, but this would result in immoderate labor. Derived from that an automated log analysis approach like the Track-Analysis-Tool is preferable, when precise durations in certain parts of the application are under investigation.

Another difference between the two approaches that did not come to mind during the implementation of the analysis tool is the interaction with anything outside of RCE. This is an inherent property of the Track-Analysis-Tool because its scope is on one specific application. When the proband tries to interact with the file system and therefore minimizes the application, this is not tracked by the prototypical implementation of the capture abilities in RCE. But even if the minimization event is in the log, all further information is inherently missing. One of the probands was searching the workflow he just created and thought that the workflow has to be opened in the file system. In the video the actions as well as the user's idea behind that was recognizable.

Moreover the analysis of the video is able to reveal very subtle problems, which probably have minor severity but still provide reasons for improvements. As a concrete example, the missing focus in one of the text fields might serve. The proband already switched to the keyboard and was about to type, hesitated because the focus was not set, moved his hand back to the mouse again, set the focus and then filled the text field. Though this is a very subtle issue and was even not memorized by the proband until the end of his task it exemplifies the idea behind this. This minor problem and the hesitation of the proband is not visible in the captured data, because the additional seconds the proband needed to set the focus could also be accounted as something different, e.g. mental operations.

Generally it is hard to detect usability problems using the Track-Analysis-Tool when a user - though she or he is confused - does not react with a lot of interaction with the interface on that, e.g. by browsing menus. The measured durations will increase but similarly to the issue mentioned before, it is not possible to differ between necessary and other mental operations. In the video it is likely that the proband verbalizes these problems. To conclude, problems that are only found by the verbalization or gestures of the proband are hard to detect in the captured data. Another example for that is the confusion a proband

had with the appearance of the different wizards. The general wizard and the wizard specific for workflows were mixed up.

Another fact to keep in mind when comparing the results of the two approaches is that they were applied simultaneously. That means, impacts inherent to a Think Aloud study also influence the metrics. For example, when the user formulates verbalizations or suggests improvement (as they often do), the duration of that specific subtask is influenced. The same holds for the hints that were given when the proband got stuck somewhere. Of course, the point in time when the hint was given can be determined from the video, but this is rather clumsy.

Generalizing the issue just mentioned it is hard to detect outer influences, that can not be captured. For example, it could happen that a colleague enters the room while the log data is captured and distracts the users. In the study the supervisor would be able to react on that, but in a remote session it is not possible to detect. Calculating idle time and take account for that in the analysis helps to cope with that, but can also be omitted, e.g. when the user unintentionally prevents the idle time counter to start by moving the mouse back and forth while she or he talks.

Moreover, the Track-Analysis-Tool is prone to over-interpretation of semantically identical sequences of actions. That means, the same semantic action can invoke different events and will thus be accounted differently in the Track-Analysis-Tool. In many cases, a human being would be able to detect the semantic meaning of the actions and not the sequence of actions itself. As a vivid example the selection of an entry at the bottom of a menu might serve. On the one hand, the user can move the mouse alongside the menu without hovering the single entries. On the other hand, the user can browse through all entries (see Figure 6.4). This ambiguity is already presented in 3.4.3—"Confusion Factor" to show that the confusion factor can be error prone.

While the first approach (a) does not invoke events, the second approach (b) invokes several arm events, although they both have the same semantic meaning.

Additionally minor flaws in the prototypical implementation of the capturing abilities can influence the results of the Track-Analysis-Tool. A minor flaw, that attracted attention during the user study and thus was not fixed to keep results comparable is described here:

To determine which text field was typed into, each text field that can possibly contribute to the success of the task is equipped with a name property. If a text field is missing this property it is assumed that the user is confused and all key events within the text field are accounted as negative events. In the implementation for one text field the name was missed. Thus, the respective subtask was rated with lots of negative events, though no actual problem was present.

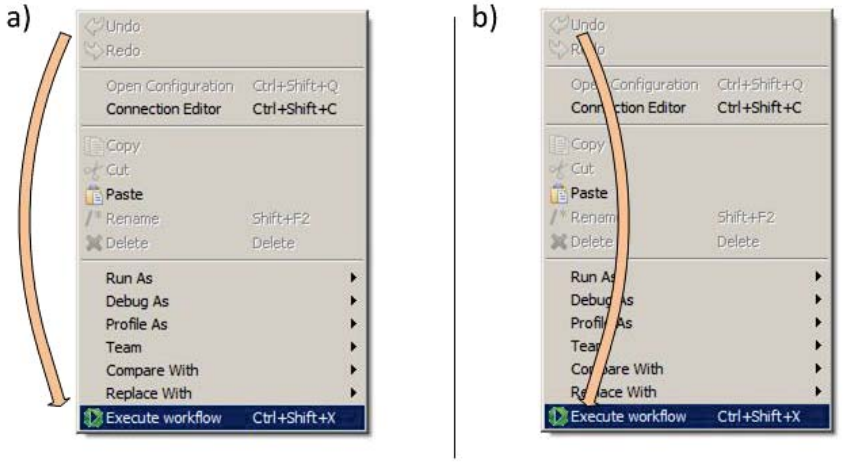

Figure 6.4: The entry "Execute Workflow" can be selected using different mouse paths

Furthermore, there are some issues one has to keep in mind using the Think Aloud method. These issues are general issues and not only limited to the scope of the thesis.

Probands often already give hints about improvement suggestions. One might tend to take that as granted and apply these improvements without further considerations, but the probands do not have the big picture of the application in mind. As the Think Aloud method reveals many even subtle problems it must always be kept in mind that the solutions to the problems must not have negative impact on the rest of the application.

6.3.2 Comparison of Identified Usability Problems

To quantify the findings mentioned above the number of found problems is reflected on from different aspects. At first, the identified problems of the first round are compared (see Figure 6.5).

The overall number of identified problems is compared. When a usability problem is observed by many probands it is counted multiple times. The Track-Analysis-Tool reveals 21 problems, while the Think Aloud approach reveals 53 problems. Taking into account the number of probands, this results in an average number of problems per proband of 3.5 with the Track-Analysis-Tool and 8.83 with the Think Aloud approach.

Another aspect is the amount of distinct problems, i.e. problems that occur several times are just accounted once in this metric. Following the numbers mentioned before, the Track-Analysis-Tool reveals 31 distinct problems while the Think Aloud approach reveals 9 distinct problems.

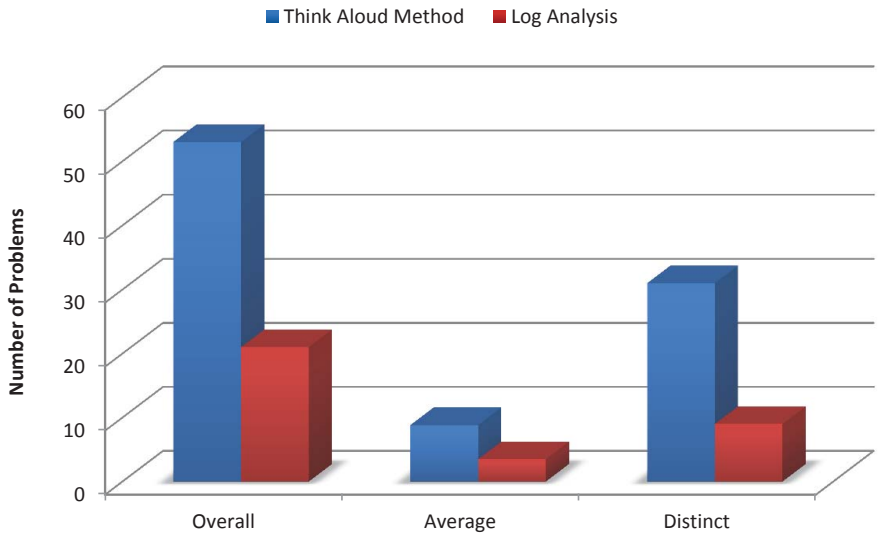

Figure 6.5: Number of found problems in round 1: The Think Aloud method reveals more problems

Comparing the heights of the bars shows that the ratio is constant to a certain degree. This shows that the Track-Analysis-Tool does not have strange behavior compared to the Think Aloud approach, like finding the same problem extra ordinary often.

In the second round 30 overall problems are revealed by the Think Aloud approach while the Track-Analysis-Tool reveals 9 problems. Per proband this leads to an average of 6 problems in the Think Aloud approach and 2.25 in the Track-Analysis-Tool. Distinct identified problems are 18 in the Think Aloud method and 6 in the Track-Analysis-Tool. As in round 1, the ratio of the numbers remains constant between the two approaches (see Figure 6.6).

Analyzing the two diagrams Figure 6.6 and Figure 6.5 shows that the ratio between the problems found in the two rounds stays constant to some extent. This can be regarded as an indicator that the log analysis produces stable results, i.e. a certain fraction of the problems is constantly found.

Figure 6.7 gives another perspective to the numbers, focusing on the comparison between the two rounds.

In addition to the overall, average-per-proband and the distinct problems from the charts above, the numbers of severe and frequent problems, as introduced in 5.6—"Problem Detection Procedure" are examined. The number of frequent problems changed from 11 in round 1 to 8 in round 2. The number of severe problems reduced from 12 in round 1 to 3 in round 2.

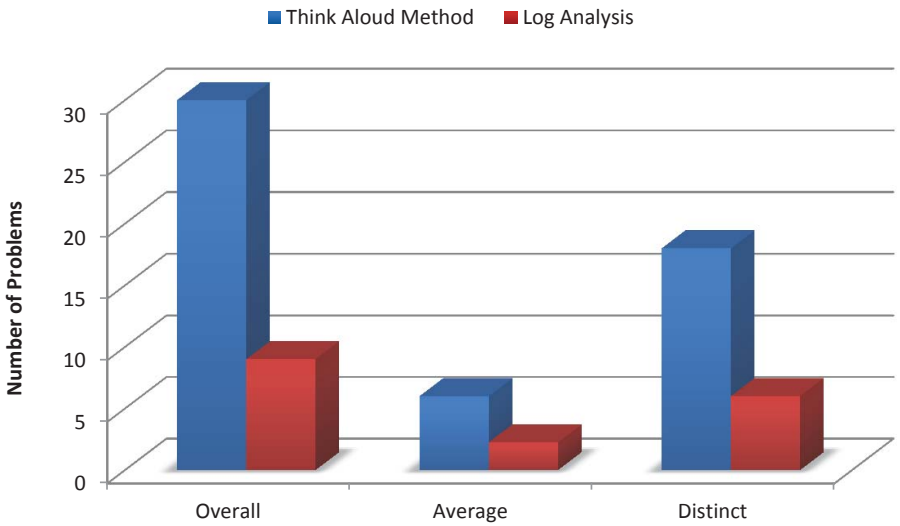

Figure 6.6: Number of found problems in round 2: The Think Aloud method still reveals more problems

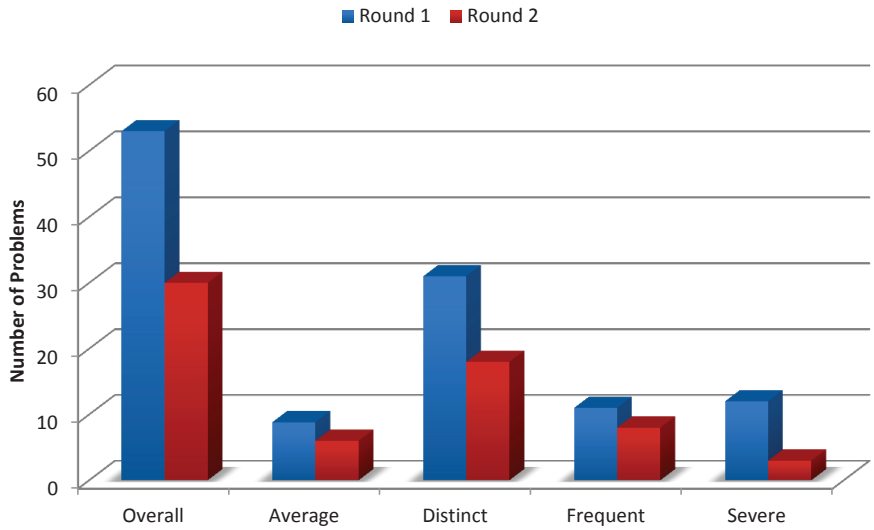

Figure 6.7: Comparison between round 1 and 2: The number of problems decreases in all aspects

These results have limitations: There were only two iterations done, the number of probands and the number of problems found is rather small. Therefore the pure numbers do not have statistical significance, but tendencies visible.

6.3.3 Comparison of Metrics

As described in 3.4.2—"Standardized Metrics" several metrics are measured in the study. To recall, the metrics cover the three aspects regarded as being vital to usability in the context of the thesis: effective-

ness, efficiency and satisfaction.

- Effectiveness: Was the task successfully executed?
- Efficiency: How fast was the task executed?
- Satisfaction: How satisfied was the user executing the task?

Effectiveness

Effectiveness is represented by the fraction of probands that succeeded in the task. All probands were able to complete the task successfully.

The amount of hints required by the probands decreased marginally from 1.67 in round 1 to 1.40 in round 2. It would be desirable, that the amount of hints decreases significantly. It is assumed that the small number of probands is the reason for that. In the second round two probands struggled with a problem, that did not appear in the first round.

Efficiency

Efficiency is represented by the durations of the single subtasks and by the execution time of the entire task.

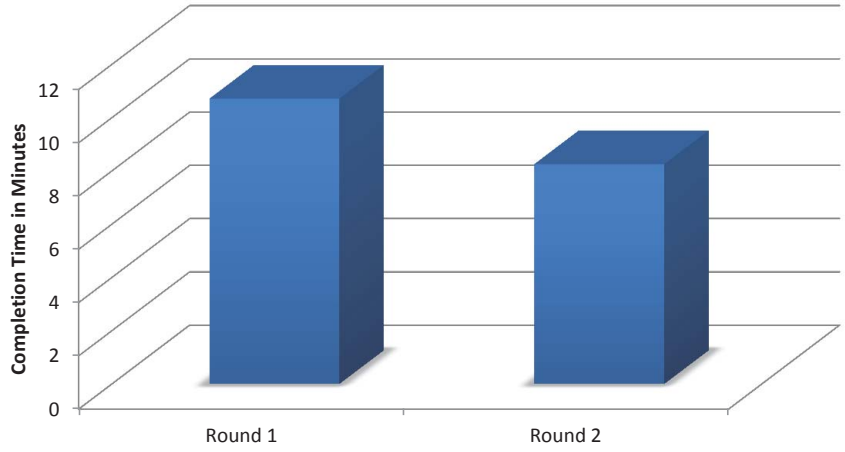

Figure 6.8: Average duration for entire task: More than 2 minutes are saved in round 2

Figure 6.8 shows that the average completion time per user decreased from round 1 to round 2 by more than 2 minutes (from 643 to 495 seconds), which is about one fifth. In the first round the completion times ranged between 388 and 945 seconds, while in the second round the completion times ranged from 336 to 689 seconds.

Taking advance of the fact that the Track-Analysis-Tool provides more quantified values, i.e. completion times per subtask, more exhaustive analysis per subtask is possible (see Figure 6.9):

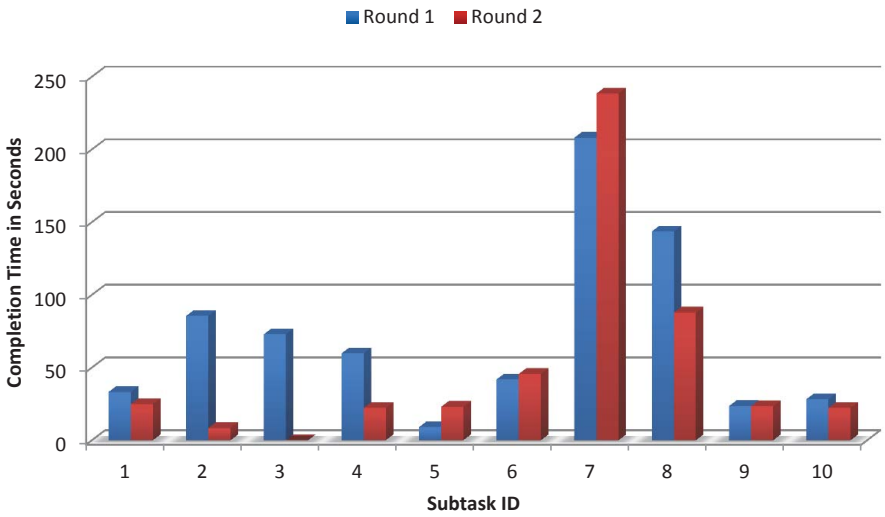

Figure 6.9: Average duration per subtask: Most times decrease, a few increase

Comparing the completion times per subtask shows that several subtasks were faster completed while some were completed slower. Reasons for that are listed per subtask:

- Subtask 1: The times to find the project wizard (or in the second round the workflow wizard) differ about 8 seconds and can be regarded as normal deviation.
- Subtask 2: In the second round the subtasks 2, 3 and 4 can be finished all in one by finishing the new designed workflow wizard. Thus, subtask 2 could be solved implicitly without spending time on it. One user invoked an event that starts subtask 2 and thus a minor duration is assigned here.
- Subtask 3: Opening a separate workflow wizard is omitted in the second round and thus the duration is zero here.
- Subtask 4: The execution of the workflow wizard improved significantly in round 2. As the red bar at subtask 4 has to be compared with the blue bars of subtask 2, 3 and 4 together it becomes obvious that the redesign of the workflow wizard enhanced the start into RCE a lot. Hints were also not required anymore within these subtasks.
- Subtask 5: The time to open the workflow in the workflow editor has slightly increased. A possible reason for that might be, that the probands in the first round had to interact with the project explorer in the course of creating a workflow. Thus, the they were familiar with the concept of the project explorer, while in the second round the project and the workflow were both created via a menu entry and the interaction with the project explorer had to be recognized first.
- Subtask 6: Adding workflow components to the workflow edi-

tor took nearly the same time in both rounds. Though further functionality was added here (adding workflow components via double-click on the palette) none of the probands in the second iteration made use of that.

- Subtask 7: The configuration of the workflow components became about 30 seconds slower in round 2. This is assumed to be induced by the small number of probands. In the second round, two probands had a problem that none of the probands of the first round faced. If the number of probands was larger, the problem is assumed to also occur in round 1 and the times would be more equal. For this subtask a design improvement was implemented that enabled the probands to open the configuration of a workflow component via the context menu of the rectangle representing the workflow component on the graphical editor. As the time for the subtask increased, the positive effect of the design improvement observed in the study was overlayed. Splitting the subtask of configuring the workflow components into two parts, i.e. opening the configuration and execution the configuration would make the effects more visible.
- Subtask 8: The times to connect the workflow components has decreased in the second iteration. This can possibly be accounted to a design improvement: The connection editor was enhances with an auto-complete functionality. That means, whenever two workflow components which are about to be connected have exactly one input and one output of the same data type, the connection is automatically pulled and the user just has to confirm it.
- Subtask 9: Opening the execution wizard worked fine in the first iteration, no changes in design were implemented and the times remained constant.
- Subtask 10: The same holds for processing the execution wizard.

Satisfaction

As mentioned in 3.4.2—"Standardized Metrics" in addition to the performance metrics covering effectiveness and efficiency the satisfaction is measured using a short questionnaire that has a look at satisfaction from several points of view. All questions are equally important and thus it is possible to average over all values. The goal of the user study was to improve performance and not to beautify the application. Nevertheless, satisfaction is important in terms of acceptance and thus considered.

The average satisfaction increased from 4.81 to 6.03 from the first to the second round. In round 1 the average ranges from 4.17 to 5.50 while in the second round the range was from 5.67 to 6.50. A graphical representation of the results is shown in Figure 6.10.

Though satisfaction is a subjective impression and thus can not be com-

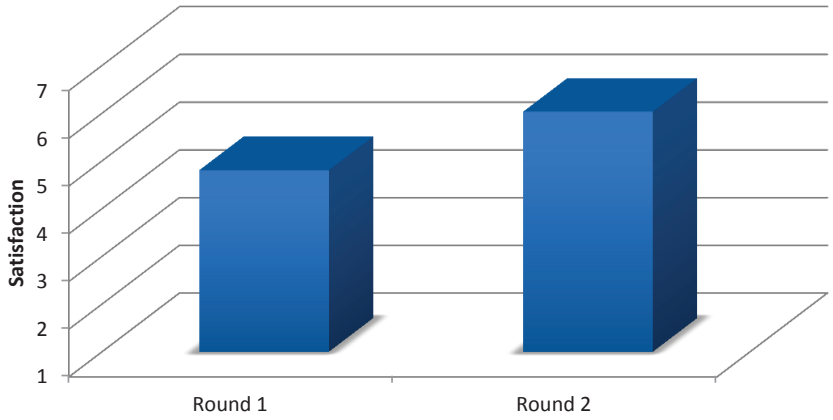

Figure 6.10: Average satisfaction per round: The satisfaction increases in round 2

pared and measured like completion times for instance, the numbers presented here match the impression of the analyst in the two rounds.

6.4 Influences of the Previous Knowledge

A second questionnaire gathering general information and prior knowledge was assessed. To recall, the overall information collects information about the age, gender, profession, employee and department. Previous knowledge is collected with regard to experience with computers, experience with the Eclipse IDE, experience with workflow-driven tools and experience with RCE.

The experience was estimated by the probands themselves, so the values are subjective. There was no correlation found between the general information or the prior knowledge on the one hand and the metrics on the other hand.

Nevertheless, collecting this data helped to get a feeling about the demographic structure of the probands under observation, as presented in 5.2.2—"Users".

6.5 Limitations of the Results and Possible Bias

There are several influences that might bias the results.

As a first bias the bridge between the languages German and English could cause confusion. RCE uses English wording. The task was given in German to provide best understandable information, because the probands have German as native language. Thus, the mapping between German and English phrases might cause trouble for some probands.

Moreover, the sample size was small and thus the results are not statistically representative. The sample size suffices to find crucial usability problems, but the resulting metrics should not be seen to strict.

Additionally the chosen probands can not be seen as representative for the entire DLR. There were only 2 departments involved and possibly departments with different focus of research might perform differently.

The user task is no real-life task. As it is artificial to use specially tailored workflow components to separate even and odd numbers instead of a script, it might appear droll to probands and influence their performance.

The used Think Aloud method can both improve or impair the users' performance, as shown in 2.2.1—"Origin and Approach".

Another bias might be the wording in the task description and in the questionnaires. People sometimes associate words with more or less positive emotions which might have an impact.

A bias that could be caused by technical issues is the assignment of events in non-linear contexts. The users are asked to complete the tasks step by step. But if a user misses a task the events could be assigned incorrectly causing corrupt metrics.

Additionally, the analyzer of the results of the Track-Analysis-Tool and the videotapes was the same person, the author of the thesis. Though it was tried, it can not be guaranteed that the analysis was not influenced to some extent.

Moreover, the satisfaction metrics and the prior knowledge was rated subjectively by the users. Though general tendencies are recognizable the given values should not be taken to strictly. Although the approach worked in the context of RCE it is necessary to apply it onto different software to verify the general applicability.

Another bias which is of more technical nature is the granularity of the subtasks. Though the subtasks as defined for the user study yielded reasonable results, it is possible that subtask definitions on different levels of abstraction would yield other - still reasonable - results.

After the discussion of the results it is presented which related work influenced the thesis.

Chapter 7

Related Work

"Shadow is a colour as light is, but less brilliant; light and shadow are only the relation of two tones."

—Paul Cezanne

The thesis encounters several areas of research. In this chapter a sample of publications is presented that is related to the thesis and had an impact on how the approach was planned and realized.

7.1 Interaction Capturing Approaches

There is several research about what is generally possible regarding logging data and analyzing.

Based on the occurrence of program errors several work was done: For example, Nielsen relates the detection and prevention of errors with the analysis of logged data as follows:

> "[A test method is] automatic computer logging of user actions. Later Analysis can determine, say, that a certain error message is issue so frequently that the "prevent errors" usability guideline should be applied to reduce the probability of the corresponding error situation." [Nie92b]

Moreover, Nielsen [Nie94b, p. 217 ff] analyses several studies that have been conducted in the context of logging data, particularly errors. As examined by Nielsen, [BMC90] conducted a study with a line-oriented system. Thereby the error messages are analyzed. The analysis showed that 30% of the errors were spelling errors, i.e. commands were probably known but just misspelled. This gives hints to add a spelling corrector to the software. Moreover, 48% of the errors were mode errors, i.e. commands were called which were not appropriate in the current

state of the system. As a consequence, larger redesigns are to be considered. Furthermore, Nielsen reports about a study [SS87] conducted by Senay and Stabler. In the study the usage of help functionality was examined. Only 10% of the help screens were invoked by 92% of the requests. Thus, the focus should be set on these 10%.

These approaches focus on the analysis of help or error messages and show that logged actions can be used to gain viable information. The focus of the thesis is not only on the detection of problems that cause an error or the usage of help functionality. The thesis aims to find general usability problems and to improve the application even in cases were an error message is not invoked despite a design worth of improvement.

Moreover, log analysis can be used to find out which features of an application are frequently used. If features are not used at all, it can be considered to discard them or to place them more present in the application.

There are several implementations dealing with logging data from applications.

Paper: Capturing and Analyzing Low-Level Events from the Code Editor

Yoon and Myers [YM11] present a plug-in for the Eclipse IDE called *Fluorite* which captures all of the low-level events when using the Eclipse code editor. In addition to the type of the event, further information like inserted or deleted text or other specific parameters are logged. Analysis and visualization tools which report various statistics about usage patterns are presented.

A short test of the plug-in showed that it works well to gather information about the user's interaction with the code editor. As the interaction to be considered in the course of the thesis goes beyond the code editor this approach is not suitable for the thesis.

Paper: NaCIN - An Eclipse Plug-In for Program Navigation-based Concern Inference

With *NaCIN* Majid and Robillard [MR05] present a plug-in for the Eclipse IDE that records how developers navigate through the code. Navigation paths and structural dependencies are analyzed and clustered into groups. This allows the detection of potential high-level concepts. The architecture and a preliminary assessment is presented in the paper. The approach presented in the paper focused on the usage of the code editor of Eclipse. The scope of the thesis is a broader one, including the entire application. Additionally the assessment is preliminary, so NaCIN is not employed in the thesis.

Paper: Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions

In the paper "Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions" Siochi and Ehrich [SE91] present a tool to analyze logged data. Basis is a log that contains all user input and all system output. Such a log can be collected automatically and non-invasively over a long period of time. The resulting data is inherently voluminous. Thus, tools and techniques that allow to detect performance and usability problems are needed.

The authors assume that repetitions are an indicator for interface problems. In their paper they present an algorithm that detects the maximal repeating pattern from a sequence of actions. The developed tool, implementing the algorithm and its application on an image-processing system is discussed. The result is, that the assumption that repetitions are an indicator for interface problems is justified and the technique is useful.

The detection of maximal repeating patterns appears to be useful in many cases. As stated in the paper, actual usability problems in the application were detected. The approach fits well to analyze very specific usability problems but fails to analyze the general aspects of the usability of interfaces.

Referring to the thesis, it is assumed that the approach is not suitable for several reasons. In a rather complex user interface as the RCE interface it is assumed to be unlikely that users will repeat the same pattern or sequence of actions multiple time. Instead, it is assumed that people browse various locations of the interface seeking for a specific functionality. Given this, the confusion factor introduced in 3.4.3—"Confusion Factor" appears to be a better indicator. In a line-oriented environment is more likely to repeat several actions as there are not so many interaction events to be invoked.

Further automated loggers named in [RCS11, p. 168] are Morae, Uservue, the Observer, Ovo Logger, Keynote and Userzoom.

7.2 Mouse Tracking

Seminar paper: Mousemap-basierte Erkennung der Problemfelder von Anwendern bei der Bedienung von Software

The seminar paper "Mousemap-basierte Erkennung der Problemfelder von Anwendern bei der Bedienung von Software" by Nelius [Nel03] describes how to setup an environment for semi-automated recording of mouse maps.

Several methods for visualizations are discussed. The interaction with

typical GUI elements like buttons or dropdown menus is recorded. The resulting mouse maps are manually analyzed and typical patterns are detected. The paper states that there is no generic approach to detect patterns. The interpretation of mouse maps always depends on the context, experience and prior knowledge.

The paper provides nice approaches to visualize mouse tracks. The analysis of the mouse maps is executed manually. As the thesis aims for an automated approach based on interaction patterns the seminar paper can only serve as a basis for further analysis.

7.3 Eye Tracking

Thesis: Eyes don't lie: Understanding Users' first impressions on website design using eye tracking

The thesis "Eyes don't lie: Understanding Users' first impressions on website design using eye tracking" by Dahal [Dah10] deals with users' first impressions on websites based on eye tracking and retrospective surveys.

It states that the first impression is important for the effectiveness of a website. The length of time it takes for users to form a first impression is analyzed and comes to a result of about 180ms. Moreover the relationship between first impression and eye movement is studied and web design factors that influence the first user impression are named.

The first user impression is just a minor part of the task to be improved, i.e. running the first successful workflow. The thesis stresses the importance of the first user impression on the overall effectiveness of a website. This is also applicable on a stand-alone application and should be kept in mind.

Paper: Identifying Web Usability Problems from Eye-Tracking Data

The paper "Identifying Web Usability Problems from Eye-Tracking Data" by Ehmke and Wilson [EW07] deals with the correlation between eye movement patterns and usability problems.

It aims to move the correlation from an anecdotal and subjective approach to a more systematic level. Therefore an initial framework is developed. A table that matches patterns and usability problems is proposed. A pattern in this context can be a single metric like a long fixation or a combination of metrics such as a scan path. The detected problems are compared to the results of other usability techniques. The comparison showed that some problems cannot be identified by eye movement analysis.

Though this paper deals with eye movement data and not with mouse

movement data, many ideas can be transferred to the thesis. The aim to identify patterns on different levels of abstraction and the comparison to other usability techniques are approaches derived from this paper.

Thesis: An eye movement analysis of web-page usability

In the thesis "An eye movement analysis of web-page usability" by Cowen [Cow01] an exploratory study was conducted to investigate the use of eye movements in evaluating usability.

Therefore eye movement measures like the average fixation time and performance measures like task completion times were recorded. Performance data revealed significant interaction between the task performed and the page it was performed on. Eye movement measures were sensitive to interaction but not significantly. Is assumed that different search strategies and behavior result in different eye movement patterns. These patterns should be further investigated.

The thesis shows that eye tracking measures do not necessarily relate significantly to performance measures. So other less expansive usability testing methods suit for the thesis.

Note: Using eye tracking for usability testing

The note "Using eye tracking for usability testing" by Namahn [Nam01] deals with the current usage, benefits and challenges of eye tracking in usability testing.

Namahn comes to the conclusion that eye tracking can help to convince managers to invest in usability testing as eye tracking results in many numbers that can be converted to useful metrics. Eye tracking in usability testing is not necessary in most applications as most usability problems can be identified with cost effective techniques. Despite that, eye tracking can quickly pay off in applications that deal with large transactions. Minimal failures due to bad usability may cause a large financial impact.

The approach of eye tracking in usability testing is relatively cost intensive compared to the budget available for the thesis. Additionally, the task to be examined in the thesis neither deals with large transactions nor with the correct placement of advertisements. It is assumed that most usability problems can be found by other methods.

7.4 Subtask Recognition

Paper: Automated segmentation of development session into task-related subsections

The paper "Automated segmentation of development session into task-related subsections" by Coman and Sillitti [CS09a] presents an approach to automatically segment streams of low level data in subtasks and discusses potential benefits.

High level information can only be collected intrusively, i.e. with an interruption of the tasks and thus influences the results. Moreover, automated collected data lacks high level information like the (sub)task the user was working on. Filtering and aggregation of low level data to higher levels can easily be done by humans but is hard for a machine. The authors propose more exhaustive testing of parameters for the algorithm as future work.

The presented approach is not tested exhaustively enough to be used without doubts. Moreover the need for an automated segmentation is not crucial because the number of subtasks is small and they have distinct and easily detectable start and end points.

7.5 From Problems to Solutions

Paper: What do Usability Evaluators Do in practice? An explorative Study of Think Aloud Testing

The paper "What do Usability Evaluators Do in practice? An explorative Study of Think Aloud Testing" by Norgaard and Hornbæk [NH06] describes the results of an explorative study of 14 Think Aloud sessions.

It states that an immediate analysis of the observations is only sporadically made. Moreover evaluators tend to seek for confirmation of already known usability problems rather than to unaffectedly discover new ones. Though these issues influence the results of sessions they are rarely discussed in literature. It appears that evaluators prioritize usability over utility.

The paper shows that the need for systematic analysis of problems exists as unstructured procedures apparently affect the results of sessions. Moreover it emphasizes that it is important to avoid biased questions or hints during a session.

Report: Analysis of usability evaluation data: An interview study with usability professionals

The study report "Analysis of usability evaluation data: An interview study with usability professionals" by Følstad, Law and Hornbæk [FLH09] examines how usability experts deal with the analysis of evaluation data collected during usability tests.

There interviews with 11 usability professionals have been conducted and analyzed. The analysis shows that only few structured procedures are used. It is an unstructured, creative process with subjective results approved by other cooperating usability experts. This methodology appears to have proven reliable.

The study report stresses the impression that the step from evaluation data to an actual usability problem is not yet established. Furthermore it shows that the step from an identified usability problem to a solution proposition remains a creative process that can hardly be formalized.

7.6 Relation of Eye Movement to Mouse Movement

Paper: Exploring How Mouse Movements Relate to Eye Movements on Web Search Results Pages

The paper "Exploring How Mouse Movements Relate to Eye Movements on Web Search Results Pages" by Rodden and Fu [RF07] examines the potential of mouse movements and their relation to eye movements.

Therefore user tests are conducted and both mouse and eye movement are recorded. Three main types of eye-mouse coordination were identified: Keeping the mouse still while reading, using the mouse as a reading aid and using the mouse to mark an interesting result. Those patterns are applied by users to varying degrees. Thus, the percentage of data points with eye and mouse position in the same region ranges from 25.8% to 59.9%.

The paper shows that it is not possible to directly map from mouse movement to eye movement. Nevertheless they can relate quite closely to each other. In the studies the mouse was often used to mark a link before clicking. This justifies the benefit of a mouse track for the problem analysis.

Having examined related work the next chapter summarized the results of the thesis and gives an overview about open issues and future work.

Chapter 8

Conclusion and Future Work

"Life is divided into three terms - that which was, which is, and which will be. Let us learn from the past to profit by the present, and from the present, to live better in the future."

—William Wordsworth

To conclude the thesis, the approach and the results are summarized and issues that remain open for future work are outlined.

8.1 What has been done?

In the course of the thesis various work has been done to gather the required information and draw appropriate conclusions.

The workflow-driven integration framework RCE was enhanced by prototypical implemented capturing abilities. The captured data is written into a JSON-based log file that contains a brought spectrum of interaction events.

An algorithm that analyses such a log file given additional context knowledge was designed and implemented. Therefore the term "subtask" is introduced. A subtask is a fraction of a task in a user study. Given additional knowledge about the beginning and end of the subtasks events can be assigned to subtasks. With additional context knowledge it is possible to rate how straightforward each subtask was accomplished. This enables an analyzer to draw conclusions about possible usability problems. As a reference another proven usability technique is determined based on literature research. The Think Aloud method was chosen and further studied.

Suitable standardized metrics were determined and the respective implementations to measure them in an automated fashion were realized. Thereby the task completion rate, completion times, the number of

given hints and satisfaction are measured. As an enhancement, application specific metrics are defined. An approach to determine when a user is confused based on unintentionally invoked events is designed and realized as a confusion warning. To depict which actions a user tried to solve a task the concept of a Distance to Success Vector has been designed.

A user study consisting of two iterations with overall 13 probands was conducted simultaneously applying the Track-Analysis-Tool and the Think Aloud method. A task covering typical interaction of users with RCE from the first start to the first successful execution of a workflow was designed. Three minimalistic workflow components tailored to the given task were implemented.

For the frequent or severe usability problems revealed in the first iteration design improvements were determined with the RCE development team using the creativity method Six Thinking Hats. A sample of the improvements was realized and integrated in the second iteration. The metrics were collected in both iterations. The metrics, the revealed usability problems and the differences between the Track-Analysis-Tool and the Think Aloud method were analyzed in detail.

8.2 What are the Discoveries?

The discoveries made in the course of the thesis refer to the realization of the conceptions and to the results of the study and the captured data.

As a first discovery temporal aspects are mentioned here. It takes a while to get the Track-Analysis-Tool running, but once set up a vast amount of data can be collected within minimal time. On the schedule the implementation was expected to take less time. In turn conducting the study and analyzing the results was expected to take longer so the schedule could still be kept.

The Track-Analysis-Tool can be applied in distributed environments and run remotely in an automated fashion. In contrast to other methods like the Think Aloud method or a field observation the analyst does not have to travel around visiting the probands. Moreover, the Track-Analysis-Tool revealed the crucial usability problems the Think Aloud method revealed, but on a higher level of abstraction. The verbalized thoughts give a deeper insight into the proband's mental model. In terms of quantification, e.g. completion times, the Track-Analysis-Tool performs very well. The Think Aloud method is able to reveal very subtle problems, sometimes not even recognized by the probands themselves. The Track-Analysis-Tool is not capable of that. Efficiently analyzing the output of the Track-Analysis-Tool requires knowledge about the domain, as otherwise names of GUI elements can not be contextualized.

Referring to a comparison between the two iterations the following dis-

coveries were made: The performance metrics improved most times due to the realized design improvements. If the performance metrics significantly decreased this can be regarded to the small sample size, so problems accidentally occurred only in the second iteration. The satisfaction when using RCE for the first time increased. The results show that the design changes improved the metrics, so actual usability problems were detected by the approaches. The determination when probands were confused while working on their task worked reliably.

8.3 What remains open for Future Work?

There are several open issues regarding the concepts and their realization.

The ability to capture the required events is prototypically implemented, but spread over various classes in a rather unclean manner. Being closely related to logging this is a typical cross-cutting concern, as it concerns various functional aspects of the application. Approaches like AOP[1] [IKL+97], reflection or the sophisticated use of extension points would allow for separation of cross-cutting concerns.

The definition of reference values such as the beginnings and ends of a subtask or the information which events are regarded as positive or negative in the context of a subtask are integrated to the analysis tool programmatically. To make the approach more flexible and easier adaptable to other applications and user tasks, a definition based on a configuration file is desirable and an open issue.

In order to enhance the statistical relevance of the results more exhaustive studies must be conducted. More probands can be considered, other applications can be tested and more iterations can be carried out until convergence of the metrics is reached.

The concept of the "Distance to Success" Vector is open to further investigation and implementation.

The analysis of the mouse track can be enhanced. The expressiveness of different visualizations of mouse tracks can be examined and possible correlations between captured interaction events and the mouse track can be explored.

Eventually it could be subject to future work how good the results of the Track-Analysis-Tool can be interpreted by an analyst who is no expert in the respective domain.

[1] AOP stands for **A**spect **O**riented **P**rogramming and is a programming paradigm that aims to increase modularity

Bibliography

[Adl10] J. Adler. *R in a Nutshell*. O'Reilly Media, 2010.

[AL08] R. Atterer, P. Lorenzi. A heatmap-based visualization for navigation within large web pages. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*. NordiCHI '08, pp. 407–410. ACM, New York, NY, USA, 2008. doi:10.1145/1463160.1463206 http://doi.acm.org/10.1145/1463160.1463206

[ASW06] E. Arroyo, T. Selker, W. Wei. Usability tool for analysis of web designs using mouse tracks. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '06, pp. 484–489. ACM, New York, NY, USA, 2006. doi:10.1145/1125451.1125557 http://doi.acm.org/10.1145/1125451.1125557

[BB84] D. C. Berry, D. E. Broadbent. On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology Section A* 36(2):209–231, 1984. doi:10.1080/14640748408402156

[BJLP02] R. Birns, J. Joffre, J. Leclerc, C. Paulsen. Getting the whole picture–The importance of collecting usability data using both concurrent think aloud and retrospective probing procedures. In *Usability Professionals Association Conference, July*. Pp. 8–12. 2002.

[BM05] R. G. Bias, D. J. Mayhew. *Cost-Justifying Usability: An Update for the Internet Age, Second Edition*. Morgan Kaufmann, April 2005.

[BMC90] J. H. Bradford, W. D. Murray, T. Carey. What kind of errors do Unix users make? In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*. Pp. 43–46. 1990.

[Bro96] J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189:194, 1996.

[Car96] W. J. Carl. Six Thinking Hats: Argumentativeness and Response to Thinking Model. March 1996.

[CDN88] J. P. Chin, V. A. Diehl, K. L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '88, pp. 213–218. ACM, New York, NY, USA, 1988. doi:10.1145/57167.57203 http://doi.acm.org/10.1145/57167.57203

[Che12] Checkstyle Development Team. Checkstyle 5.6. http://checkstyle.sourceforge.net, 2012. [Online; accessed 2013.06.24].

[Cow01] L. Cowen. An eye movement analysis of web-page usability. Master's thesis, Masters by Research in the Design and Evaluation of Advanced Interactive Systems, 2001.

[CRC12] A. Cooper, R. Reimann, D. Cronin. *About Face 3: The Essentials of Interaction Design*. Wiley, 2012. http://http://books.google.de/books?id=e75G0xIJju8C

[CS09a] I. D. Coman, A. Sillitti. Automated segmentation of development sessions into task-related subsections. *International Journal of Computers & Applications* 31(3):159, 2009. http:// citeseerx.ist.psu.edu / viewdoc / download?doi=10.1.1.188.5812&rep=rep1&type=pdf

[CS09b] I. D. Coman, A. Sillitti. Automated segmentation of development sessions into task-related subsections. *International Journal of Computers & Applications* 31(3):159, 2009.

[Dah10] S. Dahal. *Eyes Don't Lie: Understanding Users' First Impressions on Website Design Using Eye Tracking*. PhD thesis, Missouri University of Science and Technology, 2010.

[DB99] E. De Bono. *Six thinking hats*. Back Bay Books, Boston, 1999.

[DLR12] DLR. A Framework for Integration of Development Tools with Source Code Repositories. http://repoguard.tigris.org, 2012. [Online; accessed 2013.06.24].

[DLR13a] DLR. Distributed Systems and Component Software. http://dlr.de/sc/en/desktopdefault.aspx/tabid-1199/1657_read-3066, 2013. [Online; accessed 2013.01.10].

[DLR13b] DLR. DLR at a glance. http://dlr.de/dlr/en/desktopdefault.aspx/tabid-10002/#/DLR/Start/About, 2013. [Online; accessed 2013.07.12].

[DLR13c] DLR. RCE - Distributed, Workflow-driven Integration Environment. http://code.google.com/a/eclipselabs.org/p/rce, 2013. [Online; accessed 2013.01.05].

[DLR13d] DLR. Remote Component Environment (RCE). http://www.dlr.de / sc / en / desktopdefault.aspx / tabid-5625 / 9170_read-17513, 2013. [Online; accessed 2013.01.05].

[Ecl11] Eclipse Contributors. Class Composite. http://help.eclipse.org/indigo/index.jsp?topic=, 2000, 2011. [Online; accessed 2013.06.24].

[Ecl13] Eclipse Foundation. SWT: The Standard Widget Toolkit. http://eclipse.org/swt, 2013. [Online; accessed 2013.01.05].

[ES96] K. A. Ericsson, H. A. Simon. *Protocol Analysis verbal reports as data*. MIT Press, Cambridge, MA, 1996. revised edition.

[EW07] C. Ehmke, S. Wilson. Identifying web usability problems from eye-tracking data. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it*. BCS-HCI '07, pp. 119–128. British Computer Society, Swinton, UK, UK, 2007. http://dl.acm.org/citation.cfm?id=1531294.1531311

[Fas09] FasterXML. Jackson JSON Processor Wiki. http://wiki.fasterxml.com/JacksonHome, 2009. [Online; accessed 2013.06.24].

[FLH09] A. Folstad, E. Law, K. Hornbak. Analysis of usability evaluation data: An invterview study with usability professionals. September 2009. http://www.sintef.no/home/Publications/Publication/?pubid=SINTEF+A13889

[GRA10] C. Günther, A. Rozinat, W. Aalst. Activity Mining by Global Trace Segmentation. 2010. doi:10.1007/978-3-642-12186-9_13 http://dx.doi.org/10.1007/978-3-642-12186-9_13

[HB92] G. S. Hackman, D. W. Biers. Team Usability Testing: Are two Heads Better than One? *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 36(16):1205–1209, 1992. doi:10.1177/154193129203601605 http://pro.sagepub.com/content/36/16/1205.abstract

[HNZ] O. Horák, M. Novák, V. Zákoutskỳ. Heatmap Generation Techniques used for GeoWeb Application User-Interface Evaluation.

[Hol05] A. Holzinger. Usability engineering methods for software developers. *Commun. ACM* 48(1):71–74, January 2005. doi:10.1145/1039539.1039541 http://doi.acm.org/10.1145/1039539.1039541

[HS08] M. Hennig, H. Seeberger. Einführung in den Extension Point-Mechanismus von Eclipse. *JavaSPEKTRUM* 1:19–24, 2008.

[IBM07] IBM. The JFace UI framework. http://help.eclipse.org/juno/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/jface.htm, 2007. [Online; accessed 2013.07.05].

[IKL+97] J. Irwin, G. Kickzales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J. Loingtier. Aspect-oriented programming. *Proceedings of ECOOP, IEEE, Finland*, pp. 220–242, 1997.

[Jen13] Jenkins Development Team. Meet Jenkins. http://wiki.jenkins-ci.org / display / JENKINS / Meet+Jenkins, 2013. [Online; accessed 2013.06.24].

[JSO] JSON.org. Introducing JSON. http://www.json.org. [Online; accessed 2013.06.24].

[Lew82] C. Lewis. *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center, 1982.

[Lew95] J. R. Lewis. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction* 7(1):57–78, 1995. doi:10.1080/10447319509526110 http:// www.tandfonline.com / doi / abs / 10.1080 / 10447319509526110

[LFBG07] R. Lokaiczyk, A. Faatz, A. Beckhaus, M. Goertz. Enhancing just-in-time e-learning through machine learning on desktop context sensors. In *Modeling and Using Context*. Pp. 330–341. Springer, 2007.

[LG08] R. Lokaiczyk, M. Goertz. Extending low level context events by data aggregation. In *Proceedings of I-KNOW*. Volume 8, pp. 118–125. 2008.

[Lik32] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[LRV+08] E. Law, V. Roto, A. P. Vermeeren, J. Kort, M. Hassenzahl. Towards a shared definition of user experience. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '08, pp. 2395–2398. ACM, New York, NY, USA, 2008. doi:10.1145/1358628.1358693 http://doi.acm.org/10.1145/1358628.1358693

[Lun01] A. M. Lund. Measuring usability with the USE questionnaire. *Usability interface* 8(2):3–6, 2001.

[MLA10] J. McAffer, J.-M. Lemieux, C. Aniszczyk. *Eclipse Rich Client Platform*. Addison-Wesley Professional, May 2010.

[MR05] I. Majid, M. P. Robillard. NaCIN: an Eclipse plug-in for program navigation-based concern inference. In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*. Pp. 70–74. 2005.

[Nam01] Namahn. Using eye tracking for usability testing. 2001.

[Nel03] M. Nelius. Mousemap-basierte Erkennung der Problemfelder von Anwendern bei der Bedienung von Software. September 2003.

[NH06] M. Norgaard, K. Hornbaek. What do usability evaluators do in practice?: an explorative study of think-aloud testing. In *Symposium on Designing Interactive Systems: Proceedings of the 6 th conference on Designing Interactive systems*. Volume 26, p. 209–218. 2006. http://https://blog.itu.dk/DIND-F2011/files/2011/02/norgaard_dis_2006.pdf

[Nie89] J. Nielsen. The matters that really matter for hypertext usability. In *Proceedings of the second annual ACM conference on Hypertext*. HYPERTEXT '89, pp. 239–248. ACM, New York, NY, USA, 1989. doi:10.1145/74224.74244 http://doi.acm.org/10.1145/74224.74244

[Nie92a] J. Nielsen. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '92, pp. 373–380. ACM, New York, NY, USA, 1992. doi:10.1145/142750.142834 http://doi.acm.org/10.1145/142750.142834

[Nie92b] J. Nielsen. The usability engineering life cycle. *Computer* 25(3):12–22, 1992.

[Nie94a] J. Nielsen. Guerrilla HCI: Using discount usability engineering to penetrate the intimidation barrier. *Cost-justifying usability*, p. 245–272, 1994. http://www.useit.com/papers/guerrilla_hci.html

[Nie94b] J. Nielsen. *Usability Engineering*. Interactive Technologies. Elsevier Science, 1994. http://books.google.de/books?id=DBOowF7LqIQC

[Nie06] Nielsen Norman Group. F-Shaped Pattern For Reading Web Content. http://www.nngroup.com/articles/f-shaped-pattern-reading-web-content, 2006. [Online; accessed 2013.06.24].

[Not01] M. Notess. Usability, user experience, and learner experience. http://elearnmag.acm.org/archive.cfm?aid=566938, 2001. [Online; accessed 2013.06.25].

[ODR84] C. O'Malley, S. Draper, M. Riley. Constructive interaction: A method for studying human-computer-human interaction. In *PROC IFIP INTERACT'84 First Intl Conf Human-computer Interaction (London UK, 4-7 September)*. Pp. 269–274. 1984.

[OS08] J. W. Owens, S. Shrestha. How Do Users Browse a Portal Website? An Examination of User Eye Movements. *Usability News* 10(2), 2008.

[Per07] C. Perfetti. Five Survival Techniques for Creating Usable Products. http://www.uie.com/articles/product_survival_techniques, 2007. [Online; accessed 2013.06.26.

[PHR+86] K. Potosnak, P. J. Hayes, M. B. Rosson, M. L. Schneider, J. A. Whiteside. Classifying users: a hard look at some controversial issues. *SIGCHI Bull.* 17(4):84–88, April 1986. doi:10.1145/22339.22353 http://doi.acm.org/10.1145/22339.22353

[Pro10] ProContext Consulting GmbH. Usability und User Experience unterscheiden. http://blog.procontext.com/2010/03/usability-und-user-experience-unterscheiden.html, 2010. [Online; accessed 2013.06.24].

[RCS11] J. Rubin, D. Chisnell, J. Spool. *Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests*. Wiley, 2011. http://books.google.de/books?id=l_e1MmVzMb0C

[RF07] K. Rodden, X. Fu. Exploring how mouse movements relate to eye movements on web search results pages. In *SIGIR 2007 Workshop on Web Information Seeking and Interaction (WISI)*. P. 29–32. 2007. http://wattproject.com/wordpress/wp-content/uploads/2010/06/eye-mouse1.pdf

[Rub06] D. Rubel. The heart of eclipse. *Queue* 4(8):36–44, 2006.

[SB06] F. Sarodnick, H. Brau. Methoden der Usability Evaluation. *Verlag Hans Huber*, 2006.

[SB12] C. Schawel, F. Billing. Walt-Disney-Methode. In *Top 100 Management Tools*. Pp. 276–278. Springer, 2012.

[SDKP06] A. Seffah, M. Donyaee, R. B. Kline, H. K. Padda. Usability measurement and metrics: A consolidated model. *Software Quality Journal* 14(2):159–178, 2006.

[SE91] A. C. Siochi, R. W. Ehrich. Computer analysis of user interfaces based on repetition in transcripts of user sessions. *ACM Transactions on Information Systems (TOIS)* 9(4):309–335, 1991.

[Sea80] J. Seamon. *Memory & Cognition: an introduction*. Oxford University Press, 1980. http://books.google.de/books?id=Hm8QAQAAIAAJ

[Ser12] D. Sergent. Six Thinking Hats - New Tools for New Solutions. http://www.capitalquality.org/wp-content/uploads/2012/11/CQI_Six_Hats_LL_Lite1.pdf, 2012. [Online; accessed 2013.06.26.

[SFL+12] D. Seider, P. Fischer, M. Litz, A. Schreiber, A. Gerndt. Open Source Software Framework for Applications in Aeronautics and Space. In *IEEE Aerospace Conference*. 2012.

[SK05a] J. Sauro, E. Kindlund. How long should a task take? identifying specification limits for task times in usability tests. In *Proceeding of the Human Computer Interaction International Conference (HCII 2005), Las Vegas, USA*. 2005.

[SK05b] J. Sauro, E. Kindlund. A method to standardize usability metrics into a single score. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. Pp. 401–409. 2005.

[SL09] J. Sauro, J. R. Lewis. Correlations among prototypical usability metrics: evidence for the construct of usability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09, pp. 1609–1618. ACM, New York, NY, USA, 2009. doi:10.1145/1518701.1518947 http://doi.acm.org/10.1145/1518701.1518947

[SR91] B. Shackel, S. Richardson. *Human Factors for Informatics Usability*. Cambridge University Press, 1991. http://books.google.de/books?id=KSHrPgLlMJIC

[SS87] H. Senay, E. Stabler. Online Help System Usage: An Empirical Investigation. In *Proc. Second Int. Conference on Human-Computer Interaction*. 1987.

[TA10] T. Tullis, W. Albert. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Interactive Technologies. Elsevier Science, 2010. http://books.google.de/books?id=KsjpuMJ6T-YC

[The13] The Apache Software Foundation. Apache Subversion - "Enterprise-class centralized version control for the masses. http://subversion.apache.org, 2013. [Online; accessed 2013.06.24].

[VSK96] R. A. Virzi, J. L. Sokolov, D. Karis. Usability problem identification using both low- and high-fidelity prototypes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '96, pp. 236–243. ACM, New York, NY, USA, 1996. doi:10.1145/238386.238516 http://doi.acm.org/10.1145/238386.238516

[WC92] R. B. Wright, S. A. Converse. Method Bias and Concurrent Verbal Protocol in Software Usability Testing. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 36(16):1220–1224, 1992. doi:10.1177/154193129203601608 http://pro.sagepub.com/content/36/16/1220.abstract

[YM11] Y. Yoon, B. A. Myers. Capturing and analyzing low-level events from the code editor. In *Proceedings of the 3rd ACM SIGPLAN workshop on Evaluation and usability of programming languages and tools*. Pp. 25–30. 2011.

[ZS13] A. Zenkov, A. Shipilov. IOGraphica - Turn your routine work into modern art. http://iographica.com, 2013. [Online; accessed 2013.01.10].

Glossary

AOP	**A**spect **O**riented **P**rogramming, a programming paradigm that aims to increase modularity
Arm Event	An interaction event invoked whenever the mouse cursor enters a menu entry
Bundle	Unit within an application based on OSGi that encapsulates functionality
Checkstyle	A tool for static code analysis
Cognitive Walkthrough	A usability technique where an evaluator examines the product and identifies problems that novice users might face when accomplishing a given task
Component	In RCE: A configurable tool to contribute to a workflow, also called workflow component
Composite	A class of SWT used to group other SWT instances
Connection	In RCE: A channel between workflow components to exchange data
Equinox	Reference implementation of the OSGi framework
Error	In a user study: An action that prevents the user from fulfilling her or his task, thus inducing a hint by the supervisor
Event	An instance of interaction between the user and the interface
Extension Point	In RCP based applications: Elemental structure to declaratively add functionality
Eyetracking	A method so capture which parts of the screen attract particular attention
Field Observation	A usability technique where real users are observed using a product

Heuristic Evaluation A usability technique where an evaluator examines a product and judges the compliance with usability principles

GOMS **G**oals **O**perators **M**ethods and **S**election rules, a means to model human computer interaction

Hint Further information given by the supervisor to the proband when proceeding with the task is impossible otherwise

JFace Toolkit used to create graphical interfaces in RCP based applications

Jenkins A continuous integration service

Key Event An interaction event invoked whenever a keystroke is done

Likert Scale Scale to determine affirmation or rejection of a statement in a questionnaire.

Log List of events invoked when working with an application

OSGi A framework to create modular applications using a dynamic component model. Components can be installed, started, stopped, updated and uninstalled at runtime

Palette List of all tools available attached to graphical editor.

Pattern A action or set of actions when interacting with an application, e.g. a button click or filling out a form

RCP Eclipse **R**ich **C**lient **P**latform, a platform to develop rich clients in Java, using SWT and JFace as UI toolkits

Repoguard A tool to enforce specific rules on a repository

Robot Class to programmatically invoke mouse or keyboard actions

Shell A class of SWT representing windows

Six Thinking Hats Creativity method based on parallel thinking

Subtask Subunit of a task in a user test

Subversion A versioning system

SUM	**S**ingle **U**sability **M**etric, combines several metrics, e.g. task success, errors and task time into one value
SWT	**S**tandard **W**idget **T**oolkit, a graphical widget toolkit to create graphical user interfaces in Java
Think Aloud Method	Usability method where the proband is requested to verbalize her or his thoughts
Wizard	A dialog consisting of several pages, leading the user through a task step by step
Workflow	In RCE: Several workflow components exchanging and processing data to investigate a research questions

The attached disc contains:

- The exposee of the thesis
- The thesis
- The referenced websites
- The used versions of RCE
- The analysis tool
- The study and analysis results
- The sheets handed out to the probands

Appendix

A Beginning and End Conditions of the Subtasks

The following list shows the subtasks and their respective beginnings and end conditions:

1. Start the "New Project" wizard:
   - Beginning: Tracking is activated
   - End: The project or general wizard is opened
2. Execute the "New Project" wizard:
   - Beginning: Project or general wizard is opened
   - End: A project is created and appears in project explorer
3. Start the "New Workflow" wizard:
   - Beginning: A project is created and appears in project explorer
   - End: The workflow or general wizard is opened
4. Execute the "New Workflow" wizard:
   - Beginning: The workflow or general wizard is opened
   - End: A workflow is created
5. Open workflow:
   - Beginning: A workflow is created
   - End: A workflow is opened in the graphical editor
6. Add the required components:
   - Beginning: A workflow is opened in the graphical editor
   - End: All required components are added to the workflow
7. Configure the components:
   - Beginning: All required components are added to the workflow
   - End: All components are configured correctly
8. Configure required connections between components:
   - Beginning: All components are configured correctly
   - End: All connection are correctly established

9. Start the "Execute Workflow" wizard:
    - Beginning: All connection are correctly established
    - End: The workflow execution wizard is opened
10. Execute the "Execute Workflow" wizard:
    - Beginning: The workflow execution wizard is opened
    - End: The currently selected workflow is executed

B List of Typical GUI Elements and Interaction Patterns

The following list presents common graphical interface elements and typical ways to interact with them:

Button	Select
Radio button	Select
Toggle button	Toggle state
Menu	Show, hide
Menu item	Select, hover
Wizard	Open, back, next, cancel, finish, close
Tree	Expand, collapse, select
Palette	Expand, collapse, scroll
Palette item	Drag, drop, select
Checkbox	Check, uncheck
Text field	Keystroke, select
Dropdown menu	Dropdown, hide, select
Slider	Change value
Spinner	Change value
Tab folder	Select
Tab	Select, close
Scrollables	Scroll
Accordion	Expand
Canvas	Move, select, drag, drop
Tool bar	Move, select
Calender	Select, switch scope
Dialog	Open, confirm, abort
Table	Select, change order

C Excerpt from Example Results Of the Analysis

```
## Log Date: 26.04.2013 13:58:29 ##

## Analysis Date: 26.04.2013 15:29:25 ##

Overall amount of subtasks: 10
Overall duration 700.63 seconds, i.e. 11.68 minutes
Overall amount of events: 1051
Overall idle time: 134 seconds

Subtask names:
Subtask 1 : Open New (Project) Wizard
Subtask 2 : Execute New (Project) Wizard
Subtask 3 : Open New Workflow Wizard
Subtask 4 : Execute New Workflow Wizard
Subtask 5 : Open Workflow Editor
Subtask 6 : Add Components
Subtask 7 : Configure Components
Subtask 8 : Connect Components
Subtask 9 : Open Workflow Execution Wizard
Subtask 10 : Execute Workflow Execution Wizard


Subtask 1 has 6 events and took 35.1 seconds (optimal: 3 seconds) and has a rating (Positive/Neutral/Negative) of (4/2/0) without warnings and has 3 distinct events, i.e. {ArmEvent=3,
Subtask 2 has 158 events and took 219.25 seconds (optimal: 4 seconds) and has a rating (Positive/Neutral/Negative) of (17/141/0) with confusion warning and has 9 distinct events, i.e.
Subtask 3 has 111 events and took 112.65 seconds (optimal: 3 seconds) and has a rating (Positive/Neutral/Negative) of (12/99/0) with confusion warning and has 6 distinct events, i.e. {
Subtask 4 has 50 events and took 39.72 seconds (optimal: 4 seconds) and has a rating (Positive/Neutral/Negative) of (7/43/0) with confusion warning and has 9 distinct events, i.e. {Key
Subtask 5 has 23 events and took 4.71 seconds (optimal: 3 seconds) and has a rating (Positive/Neutral/Negative) of (2/21/0) without warnings and has 7 distinct events, i.e. {PartActiva
Subtask 6 has 46 events and took 31.86 seconds (optimal: 3 seconds) and has a rating (Positive/Neutral/Negative) of (6/40/0) without warnings and has 8 distinct events, i.e. {PartActiv
Subtask 7 has 220 events and took 128.82 seconds (optimal: 30 seconds) and has a rating (Positive/Neutral/Negative) of (41/175/4) without warnings and has 7 distinct events, i.e. {KeyE
Subtask 8 has 254 events and took 208.07 seconds (optimal: 10 seconds) and has a rating (Positive/Neutral/Negative) of (10/240/4) with confusion warning and has 13 distinct events, i.e
Subtask 9 has 22 events and took 26.88 seconds (optimal: 3 seconds) and has a rating (Positive/Neutral/Negative) of (1/21/0) without warnings and has 7 distinct events, i.e. {Connectio
Subtask 10 has 122 events and took 106.43 seconds (optimal: 2 seconds) and has a rating (Positive/Neutral/Negative) of (3/115/4) with confusion warning and has 6 distinct events, i.e.


---- SUBTASK 1 ---
1 - 0 - ControlEvent - - - - Tracking begins - - -
2 - 0 - ControlEvent - - - - Screen Resolution saved - - -
3 - 1 - ArmEvent - - MenuItem {&File} - File - - -
4 - 1 - ArmEvent - - MenuItem {New} - New - - -
5 - 1 - ArmEvent - - MenuItem {P&roject...} - Project... - - -
6 - 1 - WizardEvent - - - - New (project) wizard started - - -

---- SUBTASK 2 ---
1 - 0 - FocusEvent - FocusLost - Tree {} - - - - -
2 - 0 - FocusEvent - FocusGained - Composite {} - - - - - -
```

D User Task Sheet

Aufgabenstellung

Die Aufgabenstellung wird im Folgenden erläutert, wobei für jeden Unterpunkt mehrere Schritte erforderlich sein können.

1. Erstellen Sie einen Workflow mit einem Namen ihrer Wahl, z.B. "MeinWorkflow".
2. Öffnen Sie den Workflow im WorkflowEditor.
3. Fügen Sie jeweils eine Komponente vom Typ "Generator", "Separator" und "Checker" hinzu.
4. Konfigurieren Sie die Komponenten:
   - Generator:
     - Erstellen Sie ein Output vom Typ `Integer`
     - Lassen Sie Zahlen von 4 bis 22 generieren
   - Separator:
     - Erstellen Sie jeweils ein Input und ein Output vom Typ `Integer`
     - Aktivieren Sie das Trennen gerader und ungerader Zahlen
   - Checker:
     - Erstellen Sie ein Input vom Typ `Integer`
     - Aktivieren Sie das Überprüfen des Inputs
5. Verbinden Sie die Komponenten:
   - Verbindung vom Generator zum Separator
   - Verbindung vom Separator zum Checker
6. Führen Sie den Workflow aus.

Ergebnis:

- Wird der Workflow erfolgreich durchgeführt, erscheint ein Popup und der Benutzertest endet.
- Schlägt der Workflow fehl, haben Sie die Möglichkeit, es noch ein mal zu probieren, wenn Sie möchten.

E Questionnaire about Satisfaction

Zufriedenheitsumfrage RCE

Bitte bewerten Sie folgende Aussagen zur Zufriedenheit bei der Benutzung von RCE.

- Stimmen Sie der getroffenen Aussage zu, kreuzen sie bitte links an.
- Stimmen Sie der getroffenen Aussage nicht zu, kreuzen sie bitte rechts an.
- Ist Ihre Haltung neutral, kreuzen Sie bitte in der Mitte an.

Aussagen:

1. **Ich bin allgemein mit der Benutzung der Software zufrieden.**

   Stimme zu ☐—☐—☐—☐—☐—☐—☐ Stimme nicht zu

2. **Die Benutzung der Software war komfortabel.**

   Stimme zu ☐—☐—☐—☐—☐—☐—☐ Stimme nicht zu

3. **Die Oberfläche ist ansprechend gestaltet.**

   Stimme zu ☐—☐—☐—☐—☐—☐—☐ Stimme nicht zu

4. **Ich konnte einen leichten Einstieg in die Benutzung der Software finden.**

   Stimme zu ☐—☐—☐—☐—☐—☐—☐ Stimme nicht zu

5. **Die Software bietet alle Funktionen und Möglichkeiten, die ich erwartet habe.**

   Stimme zu ☐—☐—☐—☐—☐—☐—☐ Stimme nicht zu

6. **Die Informationen auf dem Bildschirm waren klar strukturiert.**

   Stimme zu ☐—☐—☐—☐—☐—☐—☐ Stimme nicht zu

Vielen Dank für Ihre Teilnahme!

F Questionnaire about Foreknowledge

Fragebogen zum Hintergrundwissen

Bitte füllen Sie folgenden Fragebogen nach eigener Einschätzung aus:

Organisatorischer Hintergrund:

1. **Wie alt sind Sie?** : ____
2. **Geschlecht:** ☐ weiblich ☐ männlich
3. **Was ist Ihr Beruf?** : _____
4. **In welchem Unternehmen arbeiten Sie?** : _____
5. **In welchem Institut/welcher Abteilung arbeiten Sie?** : _____

Technischer Hintergrund:

6. **Wie erfahren sind Sie im Umgang mit Computern allgemein?**

   Sehr erfahren ☐—☐—☐—☐—☐—☐—☐ Unerfahren

7. **Wie erfahren sind Sie in der Benutzung von Eclipse?**

   Sehr erfahren ☐—☐—☐—☐—☐—☐—☐ Unerfahren

8. **Wie erfahren sind Sie in der Benutzung von Workflow-Bearbeitungssoftware?**

   Sehr erfahren ☐—☐—☐—☐—☐—☐—☐ Unerfahren

9. **Wie erfahren sind Sie in der Benutzung von RCE?**

   Sehr erfahren ☐—☐—☐—☐—☐—☐—☐ Unerfahren

Vielen Dank für Ihre Teilnahme!

G General Information About User Task

Informationen und Aufgabenstellung

Informationen

- RCE steht für Remote Component Environment und ist eine Anwendung, um auf einfache Weise Workflows[1] aufzubauen und automatisiert auszuführen.
- Dazu werden verschiedenartige Komponenten in einem Workflow zusammengefasst, konfiguriert und verbunden.
- Ziel dieser Studie ist es, Probleme herauszufinden, auf die Benutzer bei der ersten Anwendung von RCE stoßen.
- Im Rahmen dieser Studie ist es ihre Aufgabe, einen kleinen Workflow zusammenzubauen, der typische Arbeitsschritte mit RCE abdeckt.
- Der Workflow, der im Rahmen dieser Studie zu erstellen ist, besteht aus drei Komponenten.
- Eine Komponente generiert Zahlen (Generator), eine Komponente trennt gerade und ungerade Zahlen (Separator) und eine dritte Komponente überprüft welche Zahlensequenz bei ihr ankommt (Checker).
- Sie werden während dem Benutzertest per Video aufgezeichnet. Das erleichtert die spätere Auswertung. Sollten Sie damit nicht einverstanden sein, kann auch darauf verzichtet werden.
- WICHTIG: Die Studie wird als Think-Aloud Studie durchgeführt, d.h. Sie werden gebeten ihre Gedanken laut auszusprechen.
- WICHTIG: Aus technischen Gründen ist es wichtig, zuerst alle Komponenten einzufügen, bevor Sie diese konfigurieren.

[1] Ein Workflow ist eine Zusammenstellung von sogenannten Komponenten, die Daten verarbeiten und austauschen können

H List of Frequent and Severe Usability Problems

The following list contains all severe and frequent usability problem that were detected in the first round of the user study. They are grouped into categories.

- Creation of projects and workflows:
  - It was not clear, that the created project is not the workflow
  - It was not clear, how to add a workflow to a project
  - It was not clear, the a workflow was successfully created
  - The different wizards were mixed up
  - It was not clear what the entries in the menu "new" mean
  - It was not clear, that the wizard at "new-other" is the same as "new-project" just with another filter applied
- Recognition that the workflow editor is open:
  - It was not clear, how the workflow editor looks like and when it is reached
- Add components to the workflow:
  - It was not clear, how to add components to the workflow
- Component configuration:
  - It was not clear, where the component configuration is opened
  - It was assumed, that some component options can be set by editing an endpoint
  - It was not clear, whether the entered values are saved
- Connect components:
  - It was not clear, that a connection is created, though a line was already pulled once
  - It was not clear, that a connection only exists when a line between the components is visible
  - It was not clear how connection lines are pulled
  - It was not clear, that connections don't have to be pulled from the exact edge of the tile