Automated Field Usability Evaluation Using Generated Task Trees

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
“Doctor rerum naturalium”
der Georg-August-Universität Göttingen

im Promotionsprogramm Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von

Patrick Harms
aus Göttingen

Göttingen, November 2015

Abstract

Usability is an important aspect of any kind of product. This also applies for software like desktop applications and websites, as well as apps on mobile devices and smart TVs. In a competitive market, the usability of a software becomes a discriminator between success and failure. This is especially important for software, as alternatives are often close at hand and only one click away. Hence, the software development must strive for highly usable products.

Usability engineering allows for continuously measuring and improving the usability of a software during its development and beyond. For this, it offers a broad variety of methods, that support detecting usability issues in early development stages on a prototype level, as well as during the operation of a final software. Unfortunately, most of these methods are applied manually, which increases the effort of their utilization.

In this thesis, we describe a fully automated approach for usability evaluation. This approach is a user-oriented method to be applied in the field, i.e., during the operation of a software. For this, it first traces the usage of a software by recording user actions on key stroke level. From these recordings, it compiles a model of the Graphical User Interface (GUI) of a software, as well as a usage model in the form of task trees. Based on these models and the recorded actions, our approach performs a detection of 14 different so called usability smells. These smells are exceptional user behavior and indicate usability issues. The result of the application of our approach on a software is a list of findings for each of the smells. These findings provide detailed information about the user tasks that are affected by the related usability issues, as well as about the elements of the GUI that cause the issues.

By applying it on two websites and one desktop application, we perform an in-depth validation of our approach in three case studies. In these case studies, we verify if task trees can be generated from recorded user actions and if they are representative for the user behavior. Furthermore, we apply the usability smell detection and analyze the corresponding results with respect to their validity. For this, we also compare the findings with the results of generally accepted usability evaluation methods. Finally, we conclude on the results and derive conditions for findings of our approach, which must be met to consider them as indicators for usability issues.

The results of the case studies are promising. They show, that our approach can find, fully automated, a broad range of usability issues. In addition, we show, that the findings can reference in detail the elements of the GUI that cause a usability issue. Our approach is supplemental to established usability engineering methods and can be applied with minimal effort on a large scale.

Zusammenfassung

Jedes Produkt hat eine Gebrauchstauglichkeit (Usability). Das umfasst auch Software, Webseiten und Apps auf mobilen Endgeräten und Fernsehern. Im heutigen Anbieterwettbewerb kann Usability ein entscheidender Faktor für den Erfolg eine Produktes sein. Dies gilt speziell für Software, da alternative Angebote meist schnell und einfach verfügbar sind. Daher sollte jede Softwareentwicklung Gebrauchstauglichkeit als eines ihrer Ziele definieren. Um dieses Ziel zu erreichen, wird beim Usability Engineering während der Entwicklung und der Nutzung eines Produkts kontinuierlich dessen Gebrauchstauglichkeit erfasst und verbessert. Hierfür existiert eine Reihe von Methoden, mit denen in allen Projektphasen entsprechende Probleme erkannt und gelöst werden können. Die meisten dieser Methoden sind jedoch nur manuell einsetzbar und daher kostspielig in der Anwendung.

Die vorliegende Arbeit beschreibt ein vollautomatisiertes Verfahren zur Bewertung der Usability von Software. Das Verfahren zählt zu den nutzerorientierten Methoden und kann für Feldstudien eingesetzt werden. In diesem Verfahren werden zunächst detailliert die Aktionen der Nutzer auf der Oberfläche einer Software aufgezeichnet. Aus diesen Aufzeichnungen berechnet das Verfahren ein Modell der Nutzeroberfläche sowie sogenannte Task-Bäume, welche ein Modell der Nutzung der Software sind. Die beiden Modelle bilden die Grundlage für eine anschließende Erkennung von 14 sogenannten Usability Smells. Diese definieren unerwartetes Nutzerverhalten, das auf ein Problem mit der Gebrauchstauglichkeit der Software hinweist. Das Ergebnis des Verfahrens sind detaillierte Beschreibungen zum Auftreten der Smells in den Task-Bäumen und den aufgezeichneten Nutzeraktionen. Dadurch wird ein Bezug zwischen den Aufgaben des Nutzers, den entsprechenden Problemen sowie ursächlichen Elementen der graphischen Oberfläche hergestellt.

Das Verfahren wird anhand von zwei Webseiten und einer Desktopanwendung validiert. Dabei wird zunächst die Repräsentativität der generierten Task-Bäume für das Nutzerverhalten überprüft. Anschließend werden Usability Smells erkannt und die Ergebnisse manuell analysiert sowie mit Ergebnissen aus der Anwendung etablierter Methoden des Usability Engineerings verglichen. Daraus ergeben sich unter anderem Bedingungen, die bei der Erkennung von Usability Smells erfüllt sein müssen.

Die drei Fallstudien, sowie die gesamte Arbeit zeigen, dass das vorgestellte Verfahren fähig ist, vollautomatisiert unterschiedlichste Usabilityprobleme zu erkennen. Dabei wird auch gezeigt, dass die Ergebnisse des Verfahrens genügend Details beinhalten, um ein gefundenes Problem genauer zu beschreiben und Anhaltspunkte für dessen Lösung zu liefern. Außerdem kann das Verfahren andere Methoden der Usabilityevaluation ergänzen und dabei sehr einfach auch im großen Umfang eingesetzt werden.

Acknowledgements

I would like to thank several persons who supported me in my work on this thesis. First of all, I want to thank my first supervisor Prof. Dr. Jens Grabowski. For him, it was kind of an experiment to supervise the thesis of someone with a diploma in applied sciences. I really hope, that this experiment was as successful for him as it was for me. His guidance, our discussions, and the working environment offered by his group were valuable sources of inspiration for the work.

In addition, I thank my second supervisor Prof. Dr. Dieter Hogrefe and my external reviewer Prof. Dr.-Ing. Thomas Ritz. They gave me valuable feedback and helped me to find the right structure for the work. Furthermore, I want to thank the thesis committee Prof. Dr. Carsten Damm, Prof. Dr. Wolfgang May, and Prof. Dr. Stefan Waack for investing their valuable time.

Additionally, I dedicate many thanks to my current and former colleagues in my research group and at our institute. They were good discussion partners and helped me to see things also from other points of view. Without Thomas Rings, who supported me in doing the first steps, I would not have started with this work. I also thank Ella Albrecht, Michael Göttsche, and Steffen Herbold for reviewing and commenting the thesis. And although it was hard for me to accept some comments, in the end, they were completely right and helped to make the best out of the work.

I also want to thank my family. They always motivated me go certain steps in my education, which now ended in a potential PhD. At the end of my school time, my sister said, that she hopes to call me a PhD some day. It seems, this could be the case. But she also said, that I may become a professional and popular comedian. I really hope, she is not always right.

Finally, I want to thank my cohabitee Simone Münz. She did not only spend so much time for a very detailed proof-reading and subsequent discussions, but she also supported and motivated me all the way and, especially, in the last steps of completing this work. She accepted, that I spend way too much time at work and on my desk at home, especially in the last weeks. I do not want to miss her and her support in my life.

Contents

1. Introduction

Usability of software becomes more and more important [1], as software is used in daily life also by non computer professionals. The term software herewith spans from apps on touch devices or TVs, websites, user interfaces of modern hardware, to conventional tools on desktop PCs. Usability even evolves to complex experience when using a software [2]. It, therefore, not only influences the plain usage of a software but the whole process of informing about a product, buying it, using it the first time, using it regularly, until ending the usage.

The usability of a software can be decreased by usability issues, which are aspects of the software that negatively influence its usage. To ensure a high usability of developed software, several methods established over the past years [3] that aim at detecting usability issues for a software. For example, several methods ask potential users to perform selected tasks with a software and determine the problems the users have while doing so. The detected problems can then be solved afterwards. But very often, the application of these methods requires in-depth knowledge about them and a high effort for their execution. Hence, these methods are seen as being applicable only by professionals. In addition, a prerequisite for these methods is an analysis of the tasks that users perform with a software. But these tasks are often manifold, user specific, and, hence, challenging to be analyzed. Therefore, usability evaluation should be simplified and automated [4] to scale with the increasing complexity of software and its usage scenarios. In this thesis, we describe an approach for automated usability evaluation of software incorporating an automatic detection of user tasks. This approach is intended to be applicable with minimal effort and by any person without required previous knowledge.

1.1. Motivation

Usability evaluation needs to become easier and more efficient [5] as, nowadays, more and more software are developed with a decreasing time to market. Hence, the community strives for automating usability evaluation. Automation has the advantage of cost reduction, increased consistency of uncovered usability issues and the potential to detect usability issues in system components that would not be evaluated using manual methods (e.g. because of cost and time constraints) [6, 7, 8]. In addition, automated usability evaluation can be executed rather quickly [9] without detailed previous knowledge about usability engineering. Furthermore, it allows for the objective comparison of different user interface solutions [6, 7]. Automated techniques may be applied before the system is fully implemented [6, 7] and can be used to predict the errors across a whole software design [7]. Finally, the results of automated usability evaluation methods do not underlie the human effects that come with a subjective evaluation either by end users or by experts [1]. Therefore, we strive for an automated usability evaluation method in this thesis.

Automating usability evaluations is challenging. Usually, it starts with recording users when they utilize a software. But the recorded data are large and unstructured. This makes it challenging to map the data to the actual tasks that users want to accomplish [6, 10]. This issue can be addressed by using user interface technologies that are based on models of user tasks. These models have a direct relationship to the elements of a Graphical User Interface (GUI) with which users interact. Based on this, recordings of user actions performed on a GUI can directly be transformed to recordings of user tasks. But such user interface technologies are seldom used. Hence, other approaches are required, which are directly based on existing GUI technologies without information about tasks and without requiring a developer to explicitly record the user actions or to model user tasks [10]. Due to this, our approach described in this thesis solely relies on recordings of user actions performed in GUIs. Based on the recordings, we automatically generate models of the users' tasks which we then use for an automatic usability evaluation.

Automating usability evaluation and applying it in the field can be helpful to get a big picture of the usability of a software [10]. Evaluation in the field means evaluation in the right usage context and environment [11] as well as evaluation with the real users of a software, which should be strived for [1]. Usually, the sample sizes of a field usability evaluation are large which allows for considering a large number of distinct test participants [5, 8] and usage contexts including distinct devices, times of the day, etc. [11]. This broad variety of test scenarios cannot be considered in a lab situation [12]. Hence, field evaluation takes into account a broader list of actual user requirements towards a software and supports better analyses of the actual use of the software [13]. Through this, it may also lead to other conclusions than evaluation in the lab. For example, a feature judged as important by lab participants may not be needed often in regular use [11]. For field usability evaluation, no lab is required which could interfere with normal user behavior [14]. In the field, users can do the tasks in the way they are used to it [4, 9]. In addition, field evaluation cannot be invalidated through wrong selection of test participants, wrong selection of user tasks, and disturbances through the lab setup or the evaluator [1] as no evaluator is required [9]. Therefore, our approach is intended to be applied in the field, analyzing the behavior of real users and the usability the software has for them.

An intermediate result of our approach are task trees which are a model of the usage of a software. These provide a lot of potential for subsequent usability analysis. Normally, such models are created manually which can be hard and error prone. In addition, these models may describe tasks not executed by users or executed in a different way. In our approach, we initially determine the real users' tasks automatically and then check if the evaluated system has a good usability for performing these tasks. For this, we consider often executed action combinations which are considered important and, hence, worth to be analyzed with respect to usability [15].

In general, the method described in this thesis intends to be applicable ad-hoc, easily, and without previous experience, which is so far lacking for other usability evaluation methods [16]. Instead, the current approaches for automatic usability evaluation are only supportive, for example by automating the analysis of questionnaires, performing static checks of user interface structures, and supporting the analysis of user logs [17]. Some automatic methods go one step further and focus also on task analysis. Nonetheless, they require user interface management systems which are not applicable in different contexts [10]. Our approach aims at being applicable for any kind of software independent of the underlying technology and by anybody without a high learning curve.

1.2. Scope of the Thesis

In this thesis, we present an approach for **automated field usability evaluation** based on user actions recorded during the usage of a software and based on task trees generated out of these user actions. The basic assumption for this approach is that users perform the most important actions and action combinations also most often and that these action combinations represent typical user tasks. Our major hypothesis is, that it is possible to extract user tasks from the recorded user actions and to perform an automated usability evaluation based on them. For evaluating this hypothesis, we focus on and answer the following first research question regarding the derivation of user tasks from recorded user actions:

- **RQ 1:** Can typical user tasks be determined based on recorded user actions and additional information about the structure of the GUI of a software?

This question leads to the following more detailed subquestions, which we also answer in this thesis:

- **RQ 1.1:** Which is the level of detail and semantics of the tasks that can be identified?
- **RQ 1.2:** What are requirements towards the recorded user actions (e.g., minimal number of recorded actions) and the GUI structure to allow for a detection of user tasks?
- **RQ 1.3:** Under which conditions can a detected task still be considered representative for user behavior?
- **RQ 1.4:** Can similar tasks be detected and merged and are the merge results still representative tasks?

In addition, we consider a further main research question with a focus on automating a usability evaluation based either solely on recorded user actions or on detected tasks:

- **RQ 2:** Is it possible to automatically identify usability smells, i.e., indicators for usability issues, in recorded user actions or detected user tasks with additional information about the GUI structure?

Also this question leads to more detailed subquestions answered in this thesis, which are:

- **RQ 2.1:** What are usability smell specific thresholds that should be exceeded or conditions that should be met to consider a usability smell as true hint for a usability issue?
- **RQ 2.2:** For usability smells with a direct relationship to a detected user task, which conditions should a referred task match to consider a usability smell as true positive?
- **RQ 2.3:** What are requirements towards the recorded user actions, the detected tasks, and the information about the GUI structure to allow for an effective usability smell detection?
- **RQ 2.4:** Is the detection of usability smells able to replace the application of other usability evaluation methods or does it only supplement them?

1.3. Goals and Contributions

This thesis advances the state of the art of task tree analysis and automatic usability evaluation through the following contributions:

- A general **framework for the automated field usability evaluation** which we instantiate for the scope of this thesis (Section 4.1). This framework includes the basic steps to be taken as well as the data types that need to be considered. Through this, the framework provides a basic structure for the work in this thesis but also for other automatic usability evaluation methodologies.
- An approach for the **detection of iterations and sequences** of typical user actions (Section 4.4.2) and their transformation into task trees. The result of this approach are task trees being a simple task and usage model for a software. The advantage of these structures is that they are a condensed representation of the recorded user actions and, therefore, easier to be analyzed.
- An approach for **merging similar sequences of user actions** (Section 4.4.3) to detect also task execution variants, including actions that can be left out or task execution alternatives. The resulting task trees are more condensed than the ones resulting from the previous contribution and can, hence, contribute to a better understanding of the users' tasks.
- **Support for manual task analysis** by displaying detected task trees (Section 6) or transforming them into the well-known representation of ConcurTaskTree (Annex D). Through this, software usage analysis can be simplified and task models are easier generated than through manual creation.

- A systematic **catalog of automatically detectable usability smells** (Section 4.5) being indicators for well-known usability issues. Each usability smell refers to foundations in the literature and expected user behavior, which can be searched for in the recorded data. The usability smells have a direct relation to an identified user task or to recorded user actions, as well as to the GUI of an analyzed software. Each smell provides a description that details, how the recorded actions and task trees are structured to consider a smell as present. In addition, the descriptions provide an intensity metric for each of the smells to be able to assess the severity of a detected usability smell.
- The combination of the above contributions into a **fully automated approach for usability evaluation in the field** (Section 4), which can be applied for different kinds of software including websites, desktop applications, and apps on touch devices. The approach can be utilized ad-hoc and does not require previous experience of its user. The results of applying the approach are representative for the real users of the system.

1.4. Impact

During the course of this work, intermediate results have been published in the following peer reviewed journal articles:

- P. Harms and J. Grabowski, "Usability of generic software in e-research infrastructures," *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, vol. 1, no. 3, 2011. [Online]. Available: `https://letterpress.uchicago.edu/index.php/jdhcs/article/view/89`
- P. Harms, S. Herbold, and J. Grabowski, "Extended trace-based task tree generation," *International Journal on Advances in Intelligent Systems*, vol. 7, no. 3 and 4, pp. 450–467, 12 2014. [Online]. Available: `http://www.iariajournals.org/intelligent_systems/`

In addition, the following papers have been published in peer reviewed conference proceedings:

- P. Harms, S. Herbold, and J. Grabowski, "Trace-based task tree generation," in *Proceedings of the Seventh International Conference on Advances in Computer-Human Interactions (ACHI 2014)*. XPS - Xpert Publishing Services, 2014.
- P. Harms and J. Grabowski, "Usage-based automatic detection of usability smells," in *Human-Centered Software Engineering*, ser. Lecture Notes in Computer Science, S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, and M. Winckler, Eds. Springer Berlin Heidelberg, 2014, vol. 8742, pp. 217–234. [Online]. Available: `http://dx.doi.org/10.1007/978-3-662-44811-3_13`

- P. Harms and J. Grabowski, “Consistency of task trees generated from website usage traces,” in *Proceedings of the 17th International Conference on System Design Languages (SDL Forum 2015).* Springer Berlin Heidelberg, 2015.

Furthermore, the author of this thesis has contributed to the following papers:

- S. Herbold and P. Harms, “AutoQUEST - Automated Quality Engineering of Event-driven Software,” in *Proceedings of the Fourth International Workshop on Testing Techniques & Experimentation Benchmarks for Event-Driven Software*, March 2013, pp. 134 – 139.
- S. Herbold, A. D. Francesco, J. Grabowski, P. Harms, L. M. Hillah, F. Kordon, A.-P. Maesano, L. Maesano, C. D. Napoli, F. de Rosa, M. Schneider, N. Tonellotto, M.-F. Wendland, and P.-H. Wuillemin, “The MIDAS Cloud Platform for Testing SOA Applications,” in *The 8th IEEE International Conference on Software Testing, Verification and Validation (ICST) 2015 - tools track*, Apr. 2015.

Finally, the author of this thesis supervised one student project, one bachelor thesis, and two master theses related to the scope of this work, for which he also identified and specified the topics:

- F. Trautsch, “Development and Integration of a Drupal module in a Website for gathering usage data to perform a Usability Analysis.” Student Project, Institute of Computer Science, University of Goettingen. 2013.
- F. Trautsch, “User-oriented Usability Evaluation of a Research Website.” Bachelor Thesis, Institute of Computer Science, University of Goettingen. 2013. [Online]. Available: `http://filepool.informatik.uni-goettingen.de/publication/ifi/theses/2013/ZAI-BSC-2013-14-trautsch.pdf`
- A. Deicke, “Automatic Usability Evaluation of Task Models.” Master Thesis, Institute of Computer Science, University of Goettingen. 2013. [Online]. Available: `http://filepool.informatik.uni-goettingen.de/publication/ifi/theses/2013/ZAI-MSC-2013-07-deicke.pdf`
- R. Krimmel, “Improving Automatic Task Tree Generation With Alignment Algorithms.” Master Thesis, Institute of Computer Science, University of Goettingen. 2014.

1.5. Structure of the Thesis

The thesis is structured as follows. We start with foundations in Chapter 2, in which we introduce the terminology used in this thesis. This terminology spans the usage of software (Section 2.1), the concepts of task models and trees (Section 2.2), as well as usability engineering and evaluation (Section 2.3).

In Chapter 3, we refer to scientific work related to this thesis and put our work into a broader research context. This is subdivided into existing work on automating usability evaluation in general (Section 3.1), processing of recordings of software usage with a focus on usability evaluation (Section 3.2 and 3.3), the generation of task trees (Section 3.4), and the automation of usability issue and smell detection (Section 3.5). We close Chapter 3 with a description of the research delta provided by this thesis.

In Chapter 4, we depict the details of our approach for automated field usability evaluation. We start by introducing the approach in general (Section 4.1). Then we describe, how we record user actions (Section 4.2) and derive a model of the GUI of a software (Section 4.3). Afterwards, in Section 4.4, we explain the generation of task trees from recorded user actions. The usability smell detection, subdivided into smells for task trees and smells for user actions, forms Section 4.5.

We validated our approach in three case studies. The required implementation is described in Section 5. In Section 6, we first introduce the basic setup of all case studies and provide reasons for the case study selection. Then we provide one subsection per case study (Section 6.2, 6.3, and 6.4), in which we describe and list the results of applying our approach on two websites and one desktop application. Afterwards, in Section 6.5, we briefly mention two additional experiments, which we performed in the context of this thesis. We discuss the results of the case studies in Section 7. This includes answering the research questions formulated in Section 1.2 in the sections 7.1 and 7.2, as well as showing strengths and limitations of our approach (Section 7.3). We close Section 7 with a consideration of ethical aspects. In Section 8, we conclude the thesis and provide an outlook on potential future work.

2. Foundations

This chapter introduces the foundations of this thesis which consist of terminology and basic concepts. We start by introducing terms related to GUIs and their usage. Then, we describe our notion of task trees and their structuring used throughout the thesis. Finally, we introduce usability and related concepts.

2.1. GUIs, Actions, and Events

Any software has a User Interface (UI) that can be utilized by users to interact with the software [18]. Nowadays, this interface is mostly graphical and, therefore, called Graphical User Interface (GUI). A GUI consists of many GUI elements. We subdivide GUI elements into *interaction elements*, *visual elements* and *container elements*. Interaction elements are those directly utilized by users for executing functions of a software [18]. Examples are buttons and text fields. Visual elements present information to the users but do not allow for direct user interaction. Container elements are used for structuring interaction elements, visual elements, and other container elements of a GUI. Usually, these are panels, tabbed panes, frames, or dialogs. Container elements can be in conflict with each other regarding their visibility. For example, of several sibling tabbed panes, only one can be visible at a time. We call these container elements a *view*. A view belongs to a set of views of which only one is visible at a time and which have the same parent container element. In addition, container elements can have a virtual nature in that they are not presented to the user but only used for structuring the GUI.

Container elements contain other GUI elements. Because of this relationship, GUIs follow a tree structure, that we call the *GUI model*. The leaf nodes of this tree are interaction and visual elements. The parent nodes are container elements. The root node of a GUI is a container element containing all other GUI elements directly or indirectly as its children.

A GUI model following this approach can be drawn for desktop applications, apps, and websites. An example of a GUI model for a website is shown in Figure 2.1. The root node is a virtual container element representing the whole website. Its children are also virtual and represent the individual pages of the website. These nodes are views as only one of the pages can be displayed at the same time. The other nodes refer to the Hypertext Markup Language (HTML) Document Object Model (DOM) structure of the specific page by referring to the name of the HTML tag they represent. Some of them are also virtual container elements, e.g., the node representing the *form* tag, which itself is not displayed on

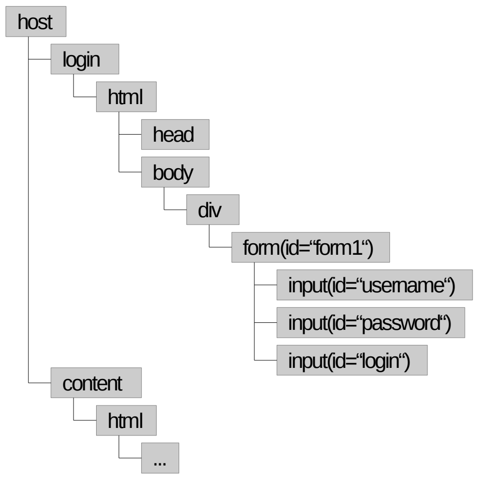


Figure 2.1.: Example of a simple GUI model.

a website. The leaf nodes of the GUI model represent interaction elements. Visual elements are not included in the example.

The following terminology is based on several papers [19, 20, 21, 22, 23] that we published in the context of our work. Interaction elements of a GUI offer to users different *actions* [18] that can be performed on a software. For example, a user can click on a button to trigger some functionality or enter a text into a text field. We refer to the set of all available actions that can be performed on a GUI of a software as the set A.

Actions can be subdivided into the two groups of *efficient* and *inefficient actions*. Efficient actions are contributing semantically to a users task. For example, entering a text into a text field can contribute to a login process. Inefficient actions are the opposite and do not contribute semantically to a users task. For example, scrolling vertically usually does not have any semantic meaning when performing a login process.

The execution of a specific action $a \in A$ by a user is called an *action instance* a'. All action instances recorded on a software belong to the set A'. An action instance a' triggers an *event* inside the software. This event signals that the user performed the respective action a and the software handles the event to process a'. An event has an *event type* and an *event target* [10]. The event type denotes the type of action the user performed such as a click, stroking a key on the keyboard, and moving the mouse. The event target refers to the GUI element on which the action was performed. Event targets are usually interaction elements. Events can also be observed on GUI elements which are no interaction elements. In this case, the corresponding action instances belong to actions which are not in A, i.e., which cannot be executed on the software. All events contain additional information, e.g., a time stamp or coordinates of a mouse click [10]. As events are representations of action instances, these information are available also for action instances.

All action instances recorded on a software can be subdivided into lists of subsequent action instances that were performed in the same view. Each list represents one opening of the view and contains the action instances that were performed when this view was displayed. For the determination of these lists, we define the function $viewActionInstances(A', view)$. The result of this function is a number of sublists resulting from the number of times the users opened the corresponding view.

The interaction of a user with a software can be seen as a language spoken by the user and understood by the software [10]. Herewith, the actions correspond to the words of the language. The combination of several actions builds sentences. Word combinations of a certain length n are named *n-grams*. We reuse the term n-gram to denote a certain combination of n actions.

2.2. Task Trees

The following terminology is based on several papers [19, 20, 21, 22, 23] that we published in the context of our work. Users perform an ordered list of actions $a_1 \dots a_n$ to reach a certain goal and, hence, to perform an individual *task* [24]. For example, users combine the actions for entering text into text fields and clicking on a confirmation button to accomplish the task of logging in on a website.

Tasks can be combined with other tasks and actions to form higher level tasks. For example, on an online shop website, the higher level task of buying a specific product is a combination of the task of logging in on the website, actions for searching and selecting the respective product, and a further task for performing the checkout. We refer to all tasks that can be performed with a software as the set T. Formally, any task $t \in T$ has an ordered list of children $c(t) = c_1 \dots c_n$. These children are either actions or other tasks, i.e., $c_i \in A \cup T \setminus \{t\}$. The number of children of a task t is defined as $|c(t)|$. Additionally, we defined that neither direct nor indirect children of a task t refer to t. This means, a task is never its own direct or indirect child.

A task is of a specific type through which it defines the execution order of its children. This order is called *temporal relationship* [25]. In our work, we consider the tasks of type *sequence*, *iteration*, *selection*, and *optional*. A sequence is a task having two or more children (i.e., $|c(t)| > 1$) which are executed in their given order. An iteration is a task that has only one child (i.e., $|c(t)| = 1$) which can be executed one or more times. A selection is a task having two or more children (i.e., $|c(t)| > 1$) of which only one can be executed. An optional is a task having only one child (i.e., $|c(t)| = 1$) which can be left out.

Through the child relationships defined above, a task forms a tree structure, that we call a *task tree*. The root node of a task tree is the task itself. The leaf nodes are the actions belonging to the task. The intermediate nodes are the child tasks belonging to the root task and define together with the root task the execution order of the actions. An example for a task tree representing a typical login procedure on a website including the entering of a user name and a password is shown in Figure 2.2. The leaf nodes are the actions that can be performed. The parent nodes, i.e., tasks, define through their type (indicated through the node name) the execution order of the actions. For example, the task *Sequence 2*, which represents the entering of the user name, is a sequence and defines that its children *Click on Text Field "username"* and *Enter Text in Text Field "username"* must be executed in their given order. *Selection 1* defines that the user may choose between entering the user name, represented through *Sequence 2*, and entering the password, represented through *Sequence 3*. *Iteration 1* defines that the user can perform this selection any amount of time. After the user name and the password are entered, the user may optionally check a check box to stay logged in, represented through *Optional 1*. Finally, the task is completed through a click on the login button.

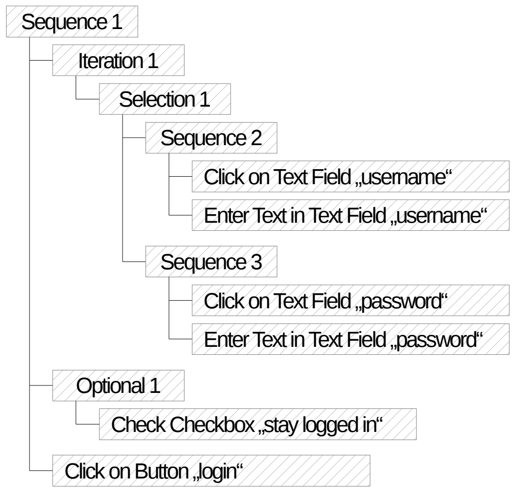

Sequence 1
Iteration 1
Selection 1
Sequence 2
Click on Text Field „username“
Enter Text in Text Field „username“
Sequence 3
Click on Text Field „password“
Enter Text in Text Field „password“
Optional 1
Check Checkbox „stay logged in“
Click on Button „login“

Figure 2.2.: Example of a simple task tree representing a login process on a website.

The execution of a task t is called a *task instance* t' [26]. A task instance also has children being task or action instances. Hence, it also forms a tree structure similar to that of a task. The leaf nodes of this tree are action instances. The root node is an instance of the respective task. The number and types of children of a task instance depend on the type of the corresponding task. The children of a *sequence instance* s' of sequence s with n children $c(s) = c_1 \dots c_n$ are a list of instances of the children of s in the same order, i.e., $c(s') = c'_1 \dots c'_n$. An *iteration instance* has one or more children all being instances of the single child of the iteration. The number of children of an iteration instance defines how often the child of the iteration was executed. A *selection instance* has exactly one child being an instance of one of the children of the selection and representing the selected execution variant. An *optional instance* has zero or one child being an instance of the single child of the optional. If the optional instance has no child, the execution of the single child of the optional was left out. Otherwise it was performed. An example for a task instance of the task tree in Figure 2.2 is shown in Figure 2.3. The nodes are instances of the respective tasks or actions. The user first enters a user name, then a password, leaves the box for staying logged in unchecked, and performs the login through a click on the respective button.

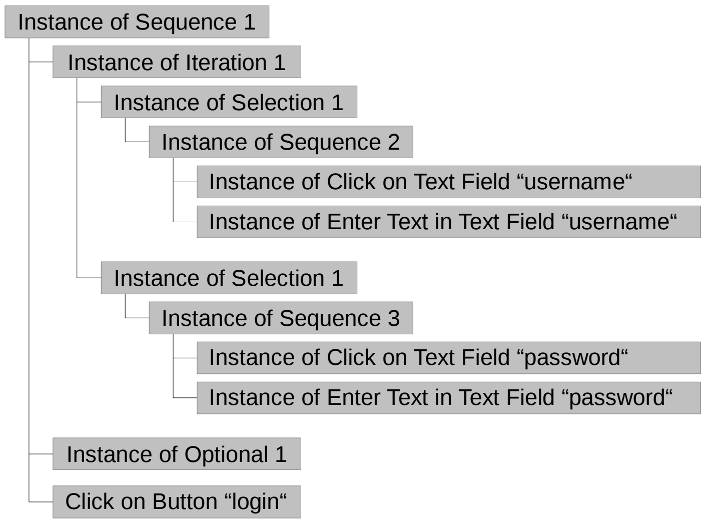


Figure 2.3.: Example of a task instance representing an execution of the task in Figure 2.2.

A task has diverse characteristics, that are of importance for our work. For example, we consider the depth of a task $depth(t)$ that we call *task depth*, which corresponds to the number of levels of the corresponding task tree. The task itself is the first level, its children the second, and so on. The actions are the last level of a task tree. The example task in Figure 2.2 has a depth of $depth(Sequence1) = 5$. In our work, we generate task trees based on recordings of action instances, i.e., events. For generated task trees, we define several functions. One is $a'(t)$ which returns all recorded action instances based on which the task t and its task tree were generated. Similarly, $a'(t')$ returns the recorded action instances which represent the instance t' of task t. A further function is $x(t)$ which returns all instances, i.e., executions, of task t.

2.3. Usability Engineering

Usability is a characteristic of products in general [27]. After ISO 9421 part 11, the usability of products focuses on executing tasks with effectiveness, efficiency and satisfaction [28]. *Effectiveness* means that the tasks are fully completed. *Efficiency* refers to the effort for task execution which should be as low as possible. *Satisfaction*, in addition, considers that the task execution must be pleasant for the users. Usability depends on the usage context, which covers user groups, tasks to be executed, as well as the physical and social environment of the user. Usability may vary strongly between different usage contexts which means, e.g., that a product can have a high usability for one person and a low usability for another.

Usability can be considered to reflect "... how easy a system is to learn and use, how productively users will be able to work and how much support users need" [29]. ISO 9126 part 1[1] also provides a definition for usability. There, the focus is on software quality and usability is, therefore, "[t]he capability of [a] software product to be understood, learned, used and attractive to the user, when used under specified conditions" [30]. Although differing in some aspects, this definition is similar to the one of ISO 9241 part 11 as it also considers the usage context, i.e., the specified conditions. In addition to effectiveness, efficiency, and satisfaction, further aspects may be considered. Examples are learnability [31], error rate [32], and attention [33].

In this thesis, we use the definition of usability after ISO 9421 part 11. Although the term is defined for products in general, in this thesis, we consider usability of software, only. The term software in this thesis covers mainly websites and GUI based computer programs on PCs. But we also consider apps on mobile devices.

A *usability issue* is a problem with the software that decreases its usability. This means, it decreases one or several factors of effectiveness, efficiency, and satisfaction [22]. Usability issues can have different causes like the visual design, the information architecture, the performance, or failures of a software. For example, a specific color combination in the visual design can make it hard for users to identify a certain GUI element and, therefore, to fulfill a task (effectiveness). Furthermore, the information architecture may require users to perform long navigation paths through a website (efficiency) to reach a specific information.

A *usability smell* in our work is exceptional behavior of users indicating one or more usability issues [22]. For example, users click on an unclickable GUI element which may indicate a usability issue with respect to the visual design. Another example is that users perform long navigation paths through a website. This indicates an inefficiency of finding a specific information, the usability issues mentioned above. A usability smell has a description of expected user behavior and refers to usability issues it may indicate. Furthermore, it has an intensity being the likelihood of indicating a usability issue.

The goal of a *usability evaluation* is to measure different aspects of the usability of a software [3], like efficiency and satisfaction. Basically, it aims at identifying usability issues. This requires the predefinition of evaluation goals, the analysis of the usage context, and finally the measurement and assessment of usability aspects using dedicated methods. The analysis of the usage context includes the identification of typical tasks users perform with a software. These tasks serve as input for the evaluation methods.

The usability evaluation methods can be subdivided into expert- and user-oriented methods [34]. Expert-oriented methods are performed by experts who know how a specific method must be applied. These methods define concrete steps the expert has to take to

[1]In the meantime, ISO 9126 is superseded by ISO 25000.

identify usability issues. For example, an expert measures the achievable efficiency of a user executing a specific task by identifying detailed actions a user has to take and then estimating the average time for the action executions [35]. In contrast, user-oriented methods follow a process in which users use a prototype or a running software for predefined tasks while they are observed by an evaluator. The observations are then analyzed and help to identify usability issues. For gathering data during the observations, different methods like taking notes, recording user actions, letting users fill out questionnaires, or *thinking aloud* [27] can be applied. Thinking aloud asks the users to verbalize their thoughts while performing the tasks so that the evaluator gets respective insights not visible from the plain actions that the users perform. User-oriented usability evaluation can be done in a laboratory or in the field [36]. A laboratory setup might influence the user and, hence, the evaluation results [27]. When done in the field, user-oriented usability evaluation lets users do their tasks in their natural environment, i.e., in the matching usage context making the results more reliable [36]. For a user-oriented usability evaluation, already three to six users are sufficient to determine the most important usability issues [34].

As *model-based usability evaluation* we refer to usability evaluation methods that utilize some kind of model. These models can describe users and the way they utilize a software [35] or the software itself [18]. For example, a model can define average durations for specific actions or it may describe the GUI. Usually, the model is created before and analyzed during the evaluation. A model can be created manually or automatically where in the latter case it is usually derived from other models (like the GUI itself) through model transformation.

The continuous application of usability evaluation methods during the development process of a software with the goal to achieve high usability of the final product is called *usability engineering* [3]. The application of the evaluation methods requires preparation. Therefore, usability engineering usually covers five tasks: a) analysis of users and context, b) modeling of a solution, c) specification of solution details, d) realization, and e) evaluation of the solution [37]. These tasks must not be understood as successive but as contributing to each other. For example, a modeling may result in requiring a further analysis of a specific aspect. Usability evaluation methods are applied in Task e) but require preparation in all other tasks.

Related to usability is the term *user experience* which covers a broader context than usability [2]. In addition to usability, it considers an "... individual's entire interaction with the [software], as well as the thoughts, feelings, and perceptions that result from that interaction." [1]. In some definitions, user experience covers a whole customer journey from searching for a product, via buying it, up to using support during usage [37]. Usability can be seen as an important part contributing to the user experience.

A further related term is *interaction design* which focuses on designing the ways users perform actions with a software [37]. Usability is also related to *accessibility* which aims at making software usable for people with certain disabilities. Finally, usability must be separated from the research field of *human computer interaction*, whose goal is to develop interaction methods with human behavior and cognitive psychology in mind [38]. This can contribute to the iterative improvement of interfaces with respect to usability.

3. Related Work

The approach described in this thesis performs an automated detection of usability smells based on recorded user actions which are transformed into task trees. As such, it is an automated, model-based, and user-oriented usability evaluation method in the field. In this chapter, we discuss the related work covering the different aspects of our approach. We start with automation in usability evaluation in general and based on GUI events. Afterwards, we cover the different aspects of our work, which are recording action instances, generating task trees, and detecting usability issues and smells.

3.1. Automation in Usability Evaluation

In 2001, Ivory and Hearst performed a survey on how automation can be introduced in usability evaluation [6]. For this, they defined a framework for usability evaluation in general. This framework consists of the three steps capture, analysis, and critique. *Capture* covers recording and pre-processing data required for a subsequent analysis. *Analysis* aims at processing and interpreting the data and identifying usability issues. Finally, *critique* proposes solutions for solving the issues. Ivory and Hearst found out that only a few usability evaluation methods can be or are automated and that automation focuses more on capture and analysis than on critique. They also propose to develop further automated methods to have a cost reduction and an improved comparability of evaluation results. The goal of our work is a full automation of usability evaluation covering capture, analysis, and also critique. In our approach, capturing is done via recording of action instances. We focus on analysis by transforming action instances into task trees and subsequently processing them for detecting usability smells. Finally, critique is provided by the resulting usability smells as they contain detailed information about the potential usability issues as well as a proposal for their solution. Through this end to end solution, our approach is applicable also for people having no deep understanding of usability and usability engineering.

Paternò and Santoro 2008 defined a framework for remote usability evaluation [39]. This framework subdivides the evaluation process into five dimensions which are 1) the interaction between the user and the evaluator, 2) the interaction platform or modality, 3) the techniques for recording user action instances, 4) the technology used for implementing a software, and 5) the type of evaluation results. In the first dimension (interaction between user and evaluator), they distinguish four different types of data collection being remote observation, remote questionnaires, critical incidents reported by the users, and automatic data collection. Our approach belongs to the last category, as we automatically record action instances without notice of the user. As Paternò and Santoro point out, this approach requires an extensive effort for the analysis of large amounts of data. In our approach, we address this by fully automating subsequent analysis steps.

The second dimension of Paternòs and Santoros framework (platform and modality), is subdivided into the three categories desktop, vocal, and mobile applications. Our work focuses on desktop applications, including websites. As websites can also be used via mobile devices, we also recorded users utilizing mobile devices in our case studies. The usability evaluation of mobile applications in the sense of Paternòs and Santoros framework, i.e., apps on mobile devices, is not yet fully supported. We only recorded action instances on an Android app and generated task trees out of them in an additional experiment. But, we did not perform a full usability evaluation based on this data. Our approach may be adapted to consider vocal interactions as action instances, as well. Through this, it may also be applicable for vocal interfaces. But in this thesis, vocal interfaces are not considered.

The third dimension of Paternòs and Santoros framework (recording action instances) is subdivided into the categories server side logging, proxy-based logging, client side logging, eye-trackers, webcam and audio recorders, as well as sensors. In our approach, we perform a client side logging for websites and desktop applications. As pointed out by Paternò and Santoro, this has several advantages, e.g., a very detailed recording of action instances. However, they also mention, without reference to other work, that usability evaluations on plain recorded user actions are unlikely to provide helpful results. In our work, we show that we gather helpful results from recorded action instances through an intermediate transformation into task trees and their subsequent analysis.

For their fourth dimension (software technology), Paternò and Santoro do not give categories but name respective technologies, e.g., Java. In our work, we record users of websites and Java applications. In an additional experiment, we recorded users of an Android app.

The fifth and final dimension of Paternòs and Santoros framework focuses on the type of evaluation result. It is subdivided into the categories task-related information, qualitative information, presentation-related data, and quantitative cognitive-physiological information. Mainly, our approach provides task-related information. However, our tasks are not predefined and premodeled as considered by Paternò and Santoro. Instead, they are generated based on action instances and, therefore, represent not intended but actual user behavior. The subsequent detection of usability smells refers to the detected tasks. We also provide presentation-related data such suboptimal positioning of GUI elements for some of the detected usability smells.

3.2. Utilizing GUI Events for Usability Evaluation

A framework that focuses on the extraction of usability information from events was defined in 2000 by Hilbert and Redmiles [10]. In their paper, they provide a classification scheme for methods that extract usability related data from recorded events, i.e., action instances. They identified five major method groups which are synchronization and searching, transformation, analysis, visualization, and integrated support. Our approach covers transformation and analysis, as well as, to some degree, visualization.

Hilbert and Redmiles subdivide the method group of *transformation* into selection, abstraction, and recoding [10]. Selection means to consider only a subset of events of interest instead of all recorded ones. An example is discarding keyboard focus change events or mouse movements. In our approach, we perform a selection of events by discarding them after recording or not even recording them. With abstraction, Hilbert and Redmiles mean combining recorded key stroke level events to higher level events. In our approach, we do this to overcome platform dependent differences between the level of recorded events. For example, on a Java platform, we record individual key press events on the keyboard which we compile to text input events on text fields whereas on websites, we directly record whole text inputs on text fields as one event. Furthermore, we apply abstraction to prevent interrelationships between distinct events, i.e., to consider any event as standing for its own. For example, a key press event on the keyboard may depend on a preceding key press event of the shift key on the keyboard. A combined text input event does not have such relationships to other events. In our approach, we also do recoding which Hilbert and Redmiles define as "... producing new event streams based on the results of selection and abstraction". When discarding events and generating higher level events from key stroke level events, the result in our work is always an adapted event stream.

The second method group of Hilbert and Redmiles that our approach belongs to is *analysis*, which they further subdivide into counts and summary statistics, sequence detection, sequence comparison, and sequence characterization. Our approach offers not only counts and summary statistics about the events, but also about the detected tasks. Our task tree generation includes a sequence detection. The methods described by Hilbert and Redmiles either detect occurrences of predefined sequence patterns or they provide general statistics about the occurrences of all sequences up to a specific length. The first method type requires a high effort to define sequences of interest as well as knowledge about the events. Its advantage is that it allows for detecting execution variants. In contrast, the second method type can not handle execution variants but can be applied without preparation. Our task tree generation does not require any manual effort for identifying interesting sequences and it also detects execution variants of similar sequences. Therefore, it directly solves the disadvantages of both variants and combines the advantages in one approach.

As sequence comparison, Hilbert and Redmiles consider approaches that compare recorded event sequences with predefined and optimal sequences or usage models. This is not supported by our approach as in our opinion, the optimal usage of a software can not be predefined by an evaluator. Instead, this predefinition represents the designers intended usage. Only the users inherently know what they consider optimal. This means, a system usage considered optimal by an evaluator may conflict with the requirements of a user. Furthermore, the predefined usage models need to be complete and, therefore, their creation usually is accompanied with a high effort for the evaluator. Our approach does not utilize predefined and considered-optimal system usage.

Finally, Hilbert and Redmiles consider sequence characterization which has the goal of deriving a usage model from events. This model is either probabilistic or grammatical. Our approach generates a grammatical model which are the task trees. In comparison to the approaches named by Hilbert and Redmiles, it does not require manual intervention and it is not sensitive to noise data.

Visualization as mentioned by Hilbert and Redmiles is used for a manual analysis of the data and requires human interpretation and, hence, respective experience and knowledge of the evaluator. This contradicts our goal of making our approach applicable for anybody without specific knowledge prerequisites. Therefore, we make only rare use of visualization. Nevertheless, our approach allows visualizing task trees and related statistics as shown in the case studies. Furthermore, we support a transformation of our task trees into other formats to support a manual inspection using existing tools. Because of the rare visualization, our approach is only partially integrated in the sense of Hilbert and Redmiles.

3.3. Recording of Action Instances

Recording of action instances on software is a well researched topic. Rosenbaum calls it behavioral data collection [13]. It can be performed in different ways, e.g., through video recordings or the creation of screencasts. In this thesis, action instances are recorded through recording GUI events.

Recording events strongly depends on the platform used for creating the GUI. In the literature, there are descriptions for recording events on Quicktime applications [40], Windows applications [6, 41], and even games [42]. Most important for this thesis is recording events on websites and Java applications.

For recording events on Java applications, usually the event handling capabilities of the respective interface technology is used. An example is recording events from Abstract Window Toolkit (AWT) GUIs where any event handled by the GUI is intercepted and logged to a log file [43]. In our approach, we practice the same which also works for Swing-based interfaces. In addition, there are techniques using aspect-oriented programming with the goal not to change the source code of a GUI for the purpose of recording. The way in which we utilize the event handling mechanisms of Swing/AWT, the source code of the GUI also does not need to be changed.

Events on websites can be recorded through a variety of techniques. Rather easily, log files of web servers can be utilized. But due to client side caching and other technologies, not all events are logged by this approach [44]. Another possibility is recording events using browser plugins. This requires an adaptation of the user's environment and is, hence, not applicable for large scale field studies [14]. Therefore, other approaches utilize only web technologies for this purpose. An example is the usage of JavaScript [14], potentially in combination with Java applets [45, 46]. For this, websites are extended with a JavaScript, which registers with the event handling mechanism of HTML on page loading to record a pre-selected set of events. If events are observed, they are either stored locally [47] or sent to a server, either through a Java applet [46] or through JavaScript [14] mechanisms. The integration of the JavaScript in the website can be done manually or automatically. For the automatic approach, a website can be routed through a proxy, which adds the JavaScript [14] to any page. Furthermore, modern Content Management Systems (CMSs) support adding a JavaScript to any page by configuration [48]. In our approach, we use only JavaScript to intercept events and to send them to a server which stores them in an eXtensible Markup Language (XML) format. This does not require any change in the user's technical setup and stores the recorded events on a location accessible for subsequent analysis.

The recording of events can also be done for apps on mobile devices. For example, Jensen and Larsen [11] recorded events on a mobile device and sent them to a server that processes the data. In addition to actions, they also recorded the startup and shutdown of an app. In an additional experiment done in this thesis, we recorded events caused by actions on an Android app and stored them on the device for later processing.

There are also technologies for setting up prototypes or whole websites that have an integrated support for recording events. For example, Remote Model-Based Evaluation (ReModEl) supports the generation of websites based on defined task models [49]. These websites are then capable of storing recorded events that have a direct relationship to the task model. In contrast to our approach, these approaches require the utilization of a specific technology and are, hence, not directly applicable on arbitrary websites.

In recent years, more and more web analytic tools are used for recording user behavior. They utilize JavaScript and other web technologies to track what users do on websites. Examples are Piwik [50] and Google Analytics [51]. Although very helpful for many research questions, these tools usually do not record user behavior as detailed as required for the approach described in this thesis. Furthermore, the recorded data may not be accessible for subsequent analysis different from that provided by the tool itself.

Recorded GUI events can have issues making a subsequent analysis difficult. For examples, events may be disordered due to the concrete recording mechanism [41]. Furthermore, event logs can become large if too many events, e.g., mouse movements, are recorded [10]. Finally, there may be inconsistencies in the logs, e.g., due to system crashes [11]. To handle all this, the recorded events require a post-processing, that we also perform in this thesis. We describe this post-processing together with our case studies, their implementation, and in detail in the Annex A.

3.4. Usage-based Generation of Task Trees

In addition to usability evaluation, we describe in this thesis an approach for automatically generating task trees based on recorded action instances. Task trees are one variant of task models which describe the nature and structure of the tasks that users perform with a software [5]. Mostly, they refer to the users' goals that can be achieved when performing a task. Following an ontology for task models defined by Van Welie et al. [24], task models are formal and they describe 1) the decomposition of tasks into subtasks and actions, 2) a tasks' flow, 3) the objects required or important for a task execution, as well as 4) the task world, i.e., the environment in which tasks are executed. The task trees that we generate in our approach focus only on task decomposition and task flow description.

Task models can be applied at several stages of the development of a software. For example, they can aid the software design, help on validating design decisions, or be used for generating task-oriented user interfaces [24]. The task trees generated in our approach can be used for a summative validation of a software and, through this, support subsequent design adaptations.

The ontology of Van Welie et al. [24] allows for comparing different variants of task models. It defines a terminology for typical concepts and their relationships used in task models. The concepts being important for our work are *task*, *basic task*, and *user action*. Van Welie et al. describe the two latter concepts as being more concrete variants of the first. A user action in Van Welie's terminology is what we simply call an action in this thesis. A basic task is "... a task for which a system provides a single function. Usually[,] basic tasks are further decomposed into user actions and system operations" [24]. With our approach, we mainly identify tasks on the level of Van Welie's basic tasks but without considering system operations. Van Welie et al. further utilize the term *unit task* which they describe "... as the simplest task that a user really wants to perform" [24]. This is the level of tasks that our approach can generate as long as sufficient users are recorded performing these tasks. However, our tasks do not refer to a user's goal as this can not be derived automatically.

In addition to the different terms for tasks, Van Welie et al. define relationships between tasks. The relationships that are generated in our approach are Van Welie's *subtask* and *trigger*. The subtask relationship corresponds to the parent child relationships in our task trees. The trigger relationship defines the order in which tasks are executed. In our approach, this is covered through the task types. Van Welie et al. mention three different trigger relationship types being *AND*, *OR*, and *NEXT*. The NEXT trigger, defining a subsequent order of tasks or actions, is covered by our task type sequence. The OR trigger, defining execution variants, is supported through our task types selection, iteration, and optional as well as their possible combinations. The AND trigger used to defined parallel task execution is not supported by our approach. According to Van Welie et al., the trigger relationship can be implemented on task level through temporal relationships or through modeling a workflow representation [24]. In our approach, we focus on the first variant which has the disadvantage that intermediate nodes may be required in the task trees to fully describe a trigger relationship [24]. The advantage is that we do not require an additional time axis which is needed for the workflow representation.

There are many different approaches utilizing tree structures for task modeling similar to our approach. These approaches usually focus on a specific utilization of the task trees.

For example, Goals, Operators, Methods, and Selection Rules (GOMS) is an approach that utilizes tree structures for manually describing a user's task with all its actions, but also with all its mental and physical effort [52]. Based on this model, an evaluator can estimate the average task execution duration. In contrast, the Task Modelling Language (TaskMODL) focuses on specifying task trees with reference to the resources required for task execution [18]. This helps to identify all required resources especially for complex tasks. Furthermore, Paternò et al. provide a notation for task trees called ConcurTaskTrees [53, 25] that allows specifying conditions, software side activities, and data flow on top of usual task structures. Furthermore, parallel task executions are supported. The task structures generated in our approach do not intend to replace any of the modeling approaches introduced by other researchers. Instead our goal is to have a simple variant of task trees which provides exactly what our approach requires or is able to generate. Through this, we hope to make our approach more understandable. Nonetheless, we show that our task tree notation can be transformed into other notations like ConcurTaskTrees.

There have been several attempts to detect tasks in recorded user actions. An example is Automated Website USability Analysis (AWUSA) [54] which tries to determine usage patterns from recorded website navigation. This work focuses on navigational patterns and does not include other actions. Furthermore, some works calculate a probabilistic model of system usage, e.g., using Petri nets [55]. These models include representations of tasks in the form of the most probable system usage. But they usually include longer action combinations which were not executed by users. In addition, these longer action combinations may have a high probability because shorter contained action combinations have a high probability. As such, these models may provide a wrong view on the actual system usage. Our task trees may also represent invalid action combinations. However, the tasks always refer to their instances which show how the tasks were effectively executed, making our model more realistic.

There are several attempts that detect tasks in form of action combinations in recorded action instances based on labeled data [4]. Labeling means to identify when a task starts or ends in a list of recorded action instances. The labels can be defined by the evaluator [56] or by the user when they start a task [57]. Other approaches try to generate the labels, for example, via time stamps of the events [58]. Afterwards, the approaches use different methods, e.g., machine learning [59] or sequence alignment [58], to identify tasks based on the labels. Our approach does not require labeled data and is as such more flexible. There are also approaches working on unlabeled data, e.g., Maximal Repeating Patterns (MRPs), which provide statistics for action combinations up to a specific length [15, 9]. This is similar to subdividing natural language texts into n-grams up to a certain length and then providing statistics for them. These approaches consider always full action combinations which may include repetitions of single actions. In our approach, we also detect repetitions of actions as standalone and include them in larger action combinations. Furthermore, MRPs do not consider shorter action combinations being part of longer action combinations as done by our approach.

A further variant for identifying tasks from user actions is programming by example. Here user actions are observed at runtime. If a pattern of actions is identified as a task, users are either proposed with next steps to be executed [60] or with automating repeated action combinations, i.e., tasks. These approaches intend to improve the efficiency of a concrete user. They do not consider the broader view of many users having different or similar tasks.

There are approaches that generate GOMS models from recorded user actions. As mentioned above, GOMS models are a hierarchical approach for task modeling similar to task trees with the goal to estimate task execution time. Examples for these approaches are ACT-R [61] and Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluations (CRITIQUE) [62]. The goal of these approaches is to ease the definition of GOMS models. Usually, an optimal task execution is performed once by an evaluator and the tools then generate a GOMS model based on this single recording of actions. A further approach called TOME generates similar models from recordings of several executions of the same task [57]. For this, the recordings are manually linked to the executed task. All of these approaches require predefined tasks whose executions are recorded. Our approach aims at detecting real user tasks and not predefining them.

The interaction of a user with a software can be seen as a language spoken by the user and understood by the software. Task trees are a grammatical description for such a language [10]. Executions of tasks are n-grams of the words belonging to the language. Therefore, approaches for grammatical inference for given language examples could be used for generating task trees based on recorded user actions. But most of the current approaches for grammatical inference require complete sentences of the language for which a grammar shall be created [63]. For natural languages, the sentences can be identified by splitting the input at, e.g., the punctuation marking the end of a sentence [64]. Considering recorded user actions, this can not be automated that easily. Instead, it would mean to label in the event streams where a task starts and where it ends. This can be time consuming especially if a large number of users is recorded. Our approach does not require labeled event streams and is, therefore, also applicable for larger data sets.

There are other attempts for grammatical inference for user actions that do not require a labeled event stream. But still, some of them require manual interaction of an evaluator [65] which is a disadvantage in comparison to our approach. To the best of our knowledge, ActionStreams is the only approach that generates grammars describing user actions without labeling the recordings [66]. This works by iterating the recorded events and if a first subsequence, i.e., n-gram, is detected that occurred a second time, then a non-terminal node for the grammar is introduced. In this approach, iterations will only be detected as a whole and not as the single elements being repeated. Through this, ActionStreams generates grammars which are quite different from our task trees. Not the actions being executed most often are subsumed to a task, but those that occurred first in their combination. Through this, their approach generates different grammars for different orders of data input. Furthermore, considering a large amount of user recordings, their approach may lead to grammar structures which include non-terminals that represent seldom executed action combinations, whereas non-terminals for often executed action combinations are missing. Our approach, instead, creates the same task trees independent of the order in which user sessions are read. The task trees are generated considering the full data set at once and not only the part that has already been read.

In parallel to the work on this thesis, there was an attempt to identify task trees having the same structure as those resulting from our approach using sequence alignment algorithms coming from the context of bio informatics [67]. These alignment algorithms are usually used to align protein sequences with each other to detect similarities. The approach is capable of detecting sequences, iterations, optionals, and selections. It was compared to our approach in terms of processing duration and the resulting task tree structures. The alignment approach is significantly slower than our approach [67]. Furthermore, a major disadvantage of the alignment approach is that the resulting task trees describe many interactions that are not possible with the recorded software. In our approach, the task trees containing only sequences and iterations (which is the case at the beginning of the generation process) do not have this downside, which allows for reliable analysis. In contrast, the final merged task trees resulting from our approach may describe invalid interactions.

3.5. Automation in Usability Issue or Smell Detection

In the literature, there are several approaches that strive for an automated detection of usability issues or smells. They can be distinguished based on the type of data they use being, e.g., GUI models, usage models, or recorded action instances. Depending on the data types, different analysis types are applied. The following paragraphs describe examples for the analysis of the different data types.

For identifying usability issues, GUI models can be checked statically against certain heuristics which are based on standards. A tool performing this is the USability Evaluation Framework (USEFul) [7], which directly evaluates websites based on their source code. Typical usability issues that can be detected through such an approach are inconsistencies regarding the structure of different pages. Other approaches analyze GUI models by calculating different metrics such as relationships between text and links or the number of images [6]. Our approach utilizes GUI models as input and focuses on the usage of a GUI. It does not perform a static check of the GUI model.

A work dealing with static GUI analysis and using the term usability smell is from Almeida et al. [68]. In their paper, the authors consider a usability smell as a less optimal structure of a GUI with respect to its usage. They define six different smells, each belonging to one of three smell categories. For example, their smell "Middle Man" belonging to the "design" category refers to unnecessary intermediate dialogues that can open while a user works in a specific view of a GUI. All the described smells focus on the static GUI structure as well as the order in which views in a GUI are shown. This is different to our definition of the term usability smell that focuses on exceptional user behavior. In addition, they base their smells on interviews of software developers making them less representative for software users.

Due to their nature, some usability issues can not easily or at all be identified automatically through static analysis of a GUI [7]. For example, a static analysis cannot identify if a GUI uses the terminology that can be understood by its users except such a terminology is predefined in a formal way. Our approach does not check a GUI model itself. Instead, we map the GUI to its usage for identifying potential usability issues. Through this, we can identify some usability issues that cannot be found based on the GUI model alone. For example, our approach aims at identifying unused GUI elements, which may not be used, because their label does not match the users terminology. But our approach cannot identify badly chosen terminology in other GUI element, e.g., in plain text.

In addition to GUI models, task models can also be the source of an automated usability issue detection. For example, they can be used to predict the efficiency of an expert user when performing a specific task as done with GOMS [69]. If task models are mapped to a GUI model, efficiency predictions can even be improved as interface specific characteristics, e.g., GUI element types and their usage, can be taken into account [70]. Task models can help to simulate user behavior on a GUI model. Through this, potential usability issues with a specific version of a GUI can be detected. Based on these findings, the GUI can be iteratively adapted to best support a set of predefined user tasks. This adaptation can be automated with the goal of reducing the remaining usability issues [71, 72]. To use all these approaches, task models need to be predefined manually. Our approach does not rely on predefined task models, but compiles them based on actual system usage. Therefore, the effort for the task model creation is lower in comparison to manual approaches. In addition, our task models could be used for the other approaches, e.g., to measure the actual user efficiency. But our task models can also become rather complex, which may make them too complex for a reuse in other approaches.

For detecting usability issues, recorded action instances, the third data type, can be analyzed through visualization. For example, they can be transformed into heat maps [40] or mapped to GUI elements [43] to show where users mostly act in a software. Furthermore, there are tools for simultaniously displaying several distinct action sequences and, through this, visualizing similarities [73]. These techniques still require a high effort for usability issue detection from an evaluator as they do not perform an automatic usability issue detection. Our approach does not simply visualize software usage, but also analyses it in an automated manner.

There is research, that performs certain statistical analysis on recorded action instances [74]. For example, one can determine, how often elements of a menu are usually used and if there are significant differences in their usage. Furthermore, a usage duration or a usage success can be visualized [11]. An analysis can also extract significant actions from recordings that may indicate usability issues such as the usage of an undo function or an erase operation [75]. These analyses provide more helpful results for an evaluator than simple visualization, which decreases the evaluator's effort. But still the evaluator has to decide if a usability issue is present by manually analyzing parts of the GUI and its usage. One aim of our approach is to minimize this requirement, and to allow for an automatic decision if a usability issue is present. This is done by calculating intensities of usability smells and identifying corresponding thresholds which need to be exceeded.

Additionally, it is possible to compare recorded action instances with expected user behavior. The expected user behavior can be an example log, that is created by an evaluator who performs a task with the software in an optimal way. This variant is applied in the tools WebHint [76], Web Automatic Usability Evaluation EnviRonment (WAUTER) [8], and Web Usability Probe [77]. The comparison can be done by a simple sequence comparison or through a more sophisticated approach using coordinate vectors on websites and their respective distances [78]. Furthermore, the expected usage can be a predefined usage model. Here, the actual usage is compared with the model. Any deviations are expected to be usability issues. Examples for these approaches with manually defined models are the different variants of the Web USer INterface Evaluator (WebRemUSINE) [79, 45, 5, 80, 12, 81] as well as similar works comparing user logs with task trees [82] in smart environments [26]. The models can also be an average system usage which is compiled from other recordings [55]. Finally, recorded action instances can be compared with a state machine of the user interface [83, 84]. The transitions in such a state machine represent the actions users are expected to take. Any deviation between recorded user actions and allowed transitions in the state machine is considered a usability issue. The comparison of logs with example logs or predefined usage models is not always easy and straightforward [10]. Furthermore, example logs and predefined usage models are a representation of the designer's intent how a software shall be used. But this intent may include usability issues for the users. In our approach, the expected usage of a system is not predefined and, therefore, not prone to contain usability issues. A downside is, that our usage models, i.e., task trees, can become rather complex and impractical to be analyzed manually.

In some setups for comparing actual with expected user behavior, it is required to do a manual mapping between recorded action instances and the predefined usage model. This increases the effort for applying these methods. In addition, the models of expected user behavior must be complete making their creation very time consuming [17] or impracticable for large systems [10]. This disadvantage can be mitigated if the expected user behavior is deducted from a model that is a byproduct of the development process. An example is ReModEl, where task models are used to generate GUIs. The usage of the interfaces can then be mapped back to the task models to identify differences. Our approach for usability smell detection does not require any predefined usage. This reduces the effort for applying our approach significantly. Furthermore, it generates task trees for seldom used parts of a GUI for which usually no models are defined manually due to the required effort. Hence, our evaluations may also find usability issues in less important GUI parts.

Some approaches search recorded action instances for specific patterns that may indicate a usability issue. For example, specific mouse movements, non-continuous up and down scrolling, or page hopping can indicate a searching behavior of the user which in turn means that the user has problems in finding a specific information [85]. These approaches resemble our work most. Especially the work of Grigera et al. [86] is to be mentioned as they also detect something they call usability smells, which is similar to our notion of the term. In their work, Grigera et al. record action instances on websites. Specific action combinations lead to so called usability threats, which are higher level actions being sent to a server and stored. Afterwards, the evaluator can request a report that contains usability smells which are in terms of Grigera et al. "problems cataloged in the literature along with the refactorings that solve them." [86]. Unfortunately, Grigera et al. do not explain the details of their usability smell detection algorithms. In comparison to our approach, Grigera's and similar approaches do not consider actual user tasks. Instead, they focus on fine grained interactions consisting of only a few actions in a specific combination. Hence, they do not have a look on the broader context in which a problem actually occurs. For example, they cannot identify if a usability issue is always existing or only during the execution of a specific user task. In our approach, we can detect for a subset of usability smells for which user task they exist. But this is not given for all usability smells.

Nonetheless, the work of Grigera et al. [86] can be the basis for a usability refactoring as described by Garrido et al. [87]. The goal of this refactoring is to provide rules to be followed and detailed steps to be taken to overcome specific usability issues. The usability issues themselves must be determined by the application of usability evaluation methods. This is similar to solving code smells using refactoring rules for source code [88], where the code smells are detected, e.g., through static source code analysis. As our approach automatically detects usability smells, it can be the basis of a subsequent refactoring. In addition, our work also uses rules similar to those of refactoring to provide detailed critique, i.e., possibilities to improve a software. But refactoring itself is out of scope of this thesis.

3.6. Summary and Research Delta

The analyzed related work shows, that automation of usability engineering is not a completely new topic. Nonetheless, Ivory and Hearst [6] argued, that in usability engineering, automation mainly strives for supporting capture and analysis required for manual evaluation methods. Additionally, they found, that automated critique, which is a prerequisite to consider a method fully automated, is mostly lacking in the attempts for automation. Also Paternò's and Santoro's [39] framework aims only at supporting manual user-oriented usability evaluation, where automation focuses on capture and analysis. Therefore, the first major research delta that we strive for in this thesis is to extend automation to also provide critique. For this, we aim at detecting usability smells. These offer critique based on the combination of their detailed description as well as tasks and GUI elements referenced by corresponding findings.

Furthermore, there are several attempts utilizing GUI events for a usability evaluation. These include techniques for synchronization and searching, transformation, analysis, visualization, and integrated support [10]. While our approach also makes use of existing techniques for transformation and supports simple visualization, its major research delta lies in the context of analysis. Here, our task tree generation allows for detecting typical action combinations, including execution variants, which results in a grammatical description of typical user behavior. Based on these task trees, we detect usability smells. Especially, the full automation, which does not require human intervention, is an aspect not provided by any other method that exists to the best of our knowledge.

This thesis does not provide a research delta considering the recording of user actions. In addition, although we utilize our own notion of task trees that matches our approach with respect to complexity and mightiness, we do not intend to define a new way of task modeling. Though, the generation of task trees based on these user recordings is brought to a new level in this thesis. In contrast to existing work, we allow for detecting task trees without required manual labeling of data or other human effort. In addition, although the generated task trees do not include user goals, they fully describe the recorded effective system usage, and no expected or possible usage as is the result of other approaches.

Finally, our inspection of related work showed several attempts for automatic detecting of usability issues. Some of these approaches also utilize recorded user actions as the basis for their analysis. We showed, that the parallel research of Grigera et al. [86] is closest to our work. But, to the best of our knowledge, there is no full description of their detected usability smells available. In addition, they do not consider detected users tasks and, hence, cannot put their findings in a broader usage context, which is a further research delta of our approach. They also did not validate, if their approach can be applied in a real environment.

4. Automated Field Usability Evaluation Using Generated Task Trees

In this chapter, we introduce our approach for automated field usability evaluation using task trees generated from action instances. We start by describing an overall framework for our approach, which we instantiate for the work in this thesis. Then, we describe the recording of action instances and the derivation of a GUI which are the prerequisites for our approach. Afterwards, we specify our task tree generation mechanism that is based on recorded action instances. Finally, we describe how we utilize recorded action instances, a GUI model, and generated task trees for a detection of usability smells.

4.1. A Framework for Automated Field Usability Evaluation

In this section, we describe the overall approach that we take for performing an automated usability evaluation. The description is twofold. We start by describing the general structure of the approach. Then, we instantiate this structure for the thesis.

4.1.1. General Structure

Automated usability evaluation is based on two types of data. The first is static data including, e.g., information about GUI elements and structure. The static data are usually prepared manually, e.g., through the definition of a model. The second is dynamic data being recorded information about the usage of the GUI by users. Partially, static data can be compiled from dynamic data, e.g., used GUI elements are part of dynamic data to have a reference to where a user action took place. The initial step for automated usability evaluation is obtaining static and dynamic data and post-processing it to match the requirements of subsequent analysis steps.

Static and especially dynamic data are typically large, what makes an analysis a challenge. Hence, as a second step the data must be compiled into one or several models describing the data. If the models are well generated, they do not only describe recorded data but are also valid for and describing unrecorded usage. A model is usually smaller and structured better than the input data. Hence, the effort for its analysis is lower. A disadvantage may be information loss as the model does not contain all information that was contained in the data anymore. On top of that, a model may be imprecise and not describe all recorded and unrecorded data. Nonetheless, in most cases a model represents an average usage of the system.

Finally, in a third step the models can be used for usability evaluation. If the models are compiled in a way that if they contain references to static and dynamic input data, this data can also be used for evaluation. The whole process covering the three steps for obtaining static and dynamic data, generating models, and analyzing the models and obtained data is shown in Figure 4.1.

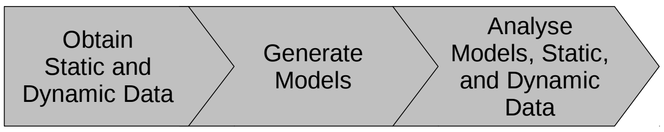


Figure 4.1.: Process for automated usability evaluation.

4.1.2. Framework Instantiation for this Thesis

In this section, we instantiate the general approach defined in the previous section. In the first step, we record action instances performed by users. This is done by recording the events caused by action instances (Section 4.2). These events also refer to GUI elements on which the actions were performed. The events are dynamic data. We do not obtain static data before applying our approach to reduce the effort of their preparation.

In the second step, we generate two types of models: a GUI model (Section 4.3) and task trees (Section 4.4). The GUI model is structured as described in Section 2.1. The task trees follow the structure described in Section 2.2. We also obtain task instances for all tasks and task trees so we know how the generated tasks were executed by the users. Through this, we have references from the models to the recorded dynamic data. These references can be used in the subsequent analysis.

In the third step, we perform a usability evaluation by detecting usability smells (Section 4.5). This includes an analysis of the recorded action instances, the derived GUI model and the generated task trees including task instances. This more concrete three-step process is show in Figure 4.2.

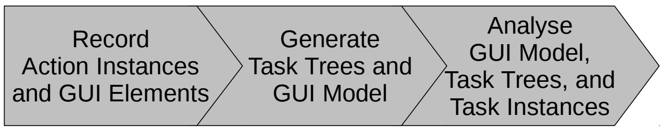


Figure 4.2.: The overall process taken in this thesis for automated usability evaluation.

4.2. Recording of Action Instances

The initial step in our overall approach is the recording of action instances and GUI elements. As described in Section 2.1, action instances cause events in a software. We record action instances by recording these events. For each action instance, exactly one event is recorded. The result is a list of events in the order they were caused by the action instances. An example for recorded action instances of a login process is shown in Figure 4.3. In the figure, the event types denote the type of the executed action. The event targets refer to the GUI elements on which the events where recorded and, hence, the actions were executed. In the example, the user starts with a click on the user name field. Then he or she enters some text and clicks the text field again to correct the text. Afterwards, the user clicks the password field and enters a password. Finally, he or she clicks the login button.

	Event Types		Event Targets
1.	Left Mouse Button Click	on	Text Field with id username
2.	Text Input „usr“	on	Text Field with id username
3.	Left Mouse Button Click	on	Text Field with id username
4.	Text Input „user“	on	Text Field with id username
5.	Left Mouse Button Click	on	Text Field with id password
6.	Text Input „“	on	Text Field with id password
7.	Left Mouse Button Click	on	Button with name „login“

Figure 4.3.: Example of recorded events caused by action instances.

The events shown in Figure 4.3 are abstract. The concrete recorded events depend on the type of platform on which the events are recorded. For example, events recorded on websites are different from those recorded for Java applications. Hence, we do a mapping to a common event meta model with the goal of harmonizing these differences. The meta model includes abstract event types for different actions which are instantiated platform specific. These abstract event types include, e.g., clicks with a mouse or entering text into a text field. The event types used in our work are listed in Table 4.1.

The event types in Table 4.1 are grouped into event types for keyboard actions, mouse actions, and complex actions. Event types for keyboard and mouse actions are very detailed and correspond to a recording on a key stroke level. Event types for complex actions combine several key stroke level events to one event and are used if the recorded platform allows for recording events on this level. For example, several subsequent events of type *Key Pressed* and *Key Released* on the same text field represent the entering of a text into the field. They can also be recorded as a single event of type *Text Input* if the respective platform supports this.

If a platform only supports recording events on key stroke level, we perform a transformation of these events into events for complex actions. For example, we transform *Key*

Abstract event type	Description	Additional information
Keyboard actions		
Key pressed	User presses key on the keyboard	Pressed key
Key released	User releases key on the keyboard	Released key
Key typed	Combination of pressing and releasing a key on the keyboard	Typed key
Mouse actions		
Mouse button down	User presses button of the mouse	Pressed mouse button
Mouse button up	User released button of the mouse	Released mouse button
Mouse button click	User clicks button of the mouse	Clicked mouse button
Complex actions		
Mouse drag & drop	User presses left button of the mouse, moves the mouse, and releases the button	Movement coordinates
Text input	User performs several key presses and releases to enter a text	Entered text
Value selection	User selects a value on a combo box or toggles a check box either through clicking or via the keyboard	Selected value
Scroll	User scrolls a panel or view in the GUI	Scrolled pixel (horizontal and vertical)

Table 4.1.: Abstract event types considered in this thesis.

Pressed and *Key Released* events into *Text Input* events. As these transformations are platform specific, they will be detailed in Section 5. Our subsequent process can handle both, key stroke level event types and complex event types. Nonetheless, to reduce the number of events processed for our overall approach and also to ensure a self-containment of the events (i.e., a single event does not have a dependency to a preceding event [10]), we strive for transforming key stroke level events into complex events before we derive GUI models and task trees.

4.3. GUI Model Derivation

The first model created in our overall process is a GUI model as shown in Figure 2.1. This model is recorded and derived together with the action instances. In this section, we describe how we determine the GUI model from the recorded events representing the action instances. This derivation is not fully identical for the different platforms on which we apply our approach. For some platforms, the GUI can directly be derived using mechanisms provided by the respective technology. For others, a more complex approach must be applied, which we describe in the following.

The targets of the recorded events are references to the GUI elements utilized in the corresponding action instances. Additionally, every GUI element has a reference to its parent container element. Using this reference, we determine a full path of a GUI element through the GUI model. This path starts with the root node of the respective GUI model and ends with the interaction element on which the event was recorded. The path is created from its end to its start, as the first known element of the path is its last element, which is the event target. An example of such a path for the interaction element *input(id="username")* in Figure 2.1 is the following. The / character separates the GUI elements from each other. The GUI elements contain additional information to be uniquely identified.

- */host/login/html/body/div/form(id="form1")/input(id="username")*

Any container element also has references to its children. Hence, we can determine GUI elements in a GUI model including those not referenced by events. For these GUI elements, we can create paths, too.

Using all paths, we generate a GUI model which is structured as described in Section 2.1. This is done iteratively through the algorithm in pseudo code shown in Algorithm 4.1. In this code, we first determine all paths for GUI elements. Then, we create an empty GUI model. Afterwards, we iteratively fill the model based on the paths. For each unseen GUI element in any of the paths, a new node is added to the GUI model. As we start from the beginnings of the paths, it is ensured that container elements are added to the model before their children are added. In the pseudo code, the model itself is a set of parent-child pairs, where the parent is a container element and the child another GUI element.

Algorithm 4.1 Creation of a GUI model out of GUI element paths from events.

```
 1: $Paths \leftarrow$ {all GUI element paths from events}
 2: $Model \leftarrow \emptyset$      // empty GUI model being a set of parent child pairs
 3: $parent \leftarrow null$
 4:
 5: for all $path \in Paths$ do
 6:    for all $guiElement \in path$ do
 7:       $Model \leftarrow Model \cup \{(parent, guiElement)\}$
 8:       $parent \leftarrow guiElement$
 9:    end for
10: end for
```

4.4. Usage-based Task Tree Generation

The second model that we create in our overall process for usability evaluation are task trees. In this section, we describe the task tree generation which utilizes the recorded action instances as input. This description was already published in a similar fashion in [20] and [21]. We start by introducing the basic steps for the task tree generation. We then describe each step in detail. Finally, we analyze the complexity of our task tree generation process.

4.4.1. Overall Process

The input for our usage-based task tree generation are recorded action instances, i.e., events. Each event represents exactly one action instance. Considering tasks and task instances as introduced in the Section 2.2, the recorded action instances are leaf nodes of instances of the tasks that we intend to detect. Hence, we generate task trees starting from the leaf nodes of the instances of the corresponding tasks.

For this, we initially transform the events into representations of action instances and store them in an ordered list, called *task instance list*, which we refer to using the variable L'. The action instances are stored in the list in the order, in which they were recorded and, hence, performed. Afterwards, we identify tasks and corresponding instances of different types. We start by detecting iterations and sequences on the task instance list. Then, we search for similar sequences and merge them. Through this, we also detect selections and optionals. In our case studies, we analyze both, task trees with unmerged and merged sequence. Therefore, the merging can also be skipped in our approach. The overall process for the task tree generation is shown in Figure 4.4. The details for the different steps of the process are described in the following subsections.

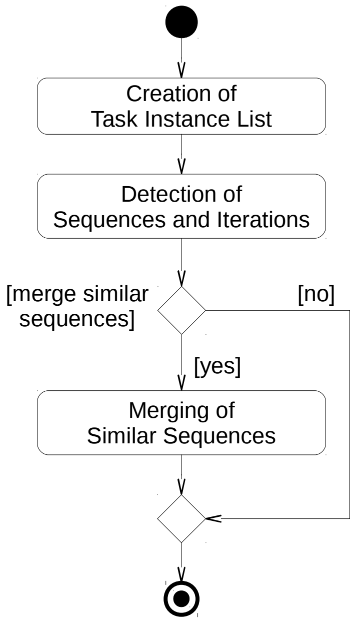


Figure 4.4.: Overall task tree generation process (adapted from [21]).

4.4.2. Iteration and Sequence Detection

The contents of this section are based on [20, 21]. The iteration and sequence detection consists of two substeps, one for detecting iterations and one for detecting sequences. The substeps are repeated alternately until neither further iterations nor sequences are detected. This interplay is shown in Figure 4.5.

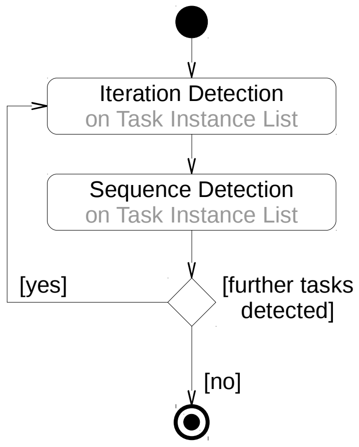


Figure 4.5.: Process for iteration and sequence detection (adapted from [21]).

The iteration and the sequence detection both search a task instance list L' for sublists of action instances that match certain criteria. This corresponds to searching a conversation in the language spoken between the user and the software for n-grams of words, i.e., actions. Therefore, we use the term n-gram to denote a sublist of L' having the length n in the following sections.

4.4.2.1. Iteration Detection

The substep for iteration detection is executed first to find iterations of actions. It searches the task instance list L' for n-grams $l'_1 \dots l'_n$ of instances of the same action, e.g., two subsequent clicks on the same button. The n-grams do not have a fixed length. If it finds such n-grams for a specific action a, it

1. creates an iteration i having a as its single child,
2. identifies the set of all n-grams $l'_1 \dots l'_n$ of the task instance list containing only one or more subsequent instances of a,
3. for each n-gram $l'_i \in l'_1 \dots l'_n$, creates an instance i' of i,

4. adds the instances of a in l_i' as children to i', and
5. replaces l_i' with the corresponding iteration instance i' in the task instance list.

As a result, the task instance list contains action instances as well as iteration instances. An example for the iteration detection is shown in Figure 4.6 a and 4.6 b. Figure 4.6 a shows an initial task instance list with action instances. The respective actions are indicated through the letters in the grey boxes. Thus, instances using the same letter represent instances of the same action. In the example, the iteration detection identifies two separated n-grams of subsequent instances of Action *b* (indicated through the dotted boxes). For both n-grams, it creates one iteration *Iteration 1* as well as two instances of *Iteration 1* that replace the two n-grams in the task instance list (Figure 4.6 b). The detected iteration is shown in the upper part of Figure 4.6 d.

If the iteration detection identifies repetitions of several actions, it creates iterations and respective instances for all of them and performs also the corresponding replacements in

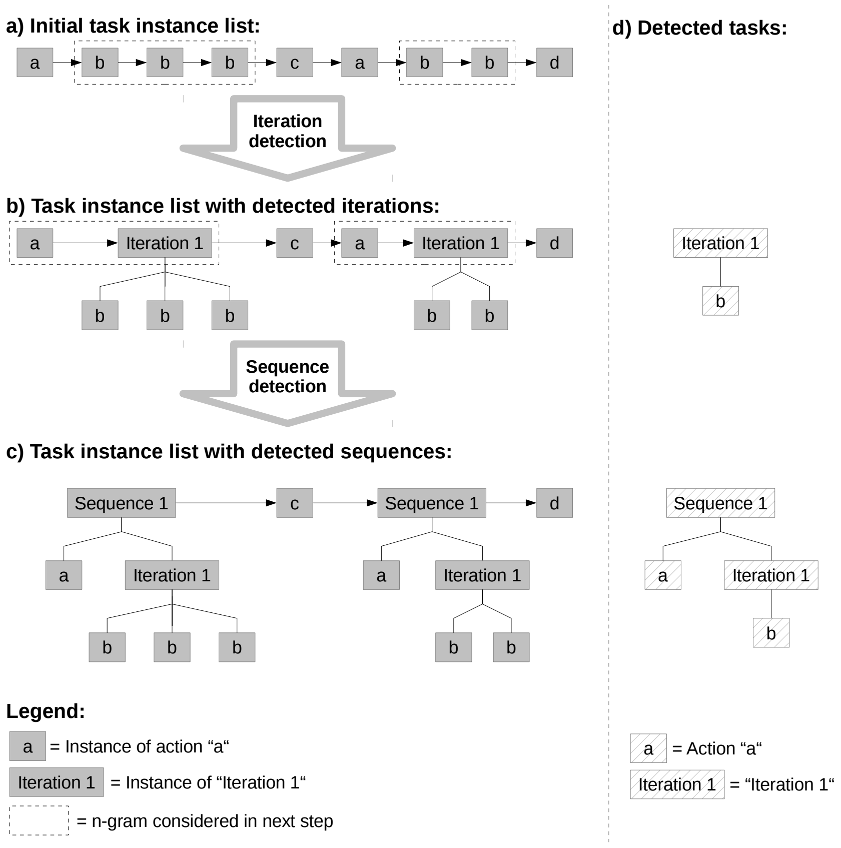

a) Initial task instance list:
a b b b c a b b d
Iteration detection
b) Task instance list with detected iterations:
a Iteration 1 c a Iteration 1 d
b b b b b
Sequence detection
c) Task instance list with detected sequences:
Sequence 1 c Sequence 1 d
a Iteration 1 a Iteration 1
b b b b b
d) Detected tasks:
Iteration 1
b
Sequence 1
a Iteration 1
b
Legend:
a = Instance of action "a"
Iteration 1 = Instance of "Iteration 1"
= n-gram considered in next step
a = Action "a"
Iteration 1 = "Iteration 1"

Figure 4.6.: Example for iteration and sequence detection (adapted from [20, 21]).

the task instance list. There is no predefined order for the replacements as the n-grams do not overlap and can be replaced independently. Through its nature, the iteration detection adapts the task instance list and, at the same time, ensures that it still represents the original order of executed actions.

4.4.2.2. Sequence Detection

After the iteration detection, we perform the substep for sequence detection. The sequence detection searches the task instance list L' for identical n-grams $l'_1 \dots l'_n$ having a minimum length of two. Identical means that

- the n-grams have the same length ($|l'_1| = |l'_2| = \dots = |l'_n|$) and
- at each position the n-grams consist either of instances of the same action or instances of the same task, e.g., iteration.

If the algorithm detects identical n-grams $l'_1 \dots l'_n$ it

1. creates a sequence s having actions or tasks represented by the n-grams as its children,
2. for each n-gram $l'_i \in l'_1 \dots l'_n$, creates an instance s'_i of s,
3. adds child instances to s'_i being the action or task instances belonging to the n-gram l'_i represented through s'_i, and
4. replaces each n-gram l'_i with the corresponding s'_i in the task instance list.

As a result, the task instance list contains action, iteration, and sequence instances. Similar to the iteration detection, the sequence detection adapts the task instance list and, yet, ensures its order and representativeness for the executed actions. An example for the sequence detection is shown in Figure 4.6 b and 4.6 c. Figure 4.6 b shows a task instance list after an iteration detection with action and iteration instances. In the example, the sequence detection identifies two identical n-grams consisting of instances of Action a and instances of *Iteration 1* (indicated through the dotted boxes). For both n-grams it creates one sequence *Sequence 1* as well as two instances of *Sequence 1* that replace the two n-grams in the task list (Figure 4.6 c). The detected sequence is shown in the middle part of Figure 4.6 d.

The sequence detection may identify several sets of identical n-grams. For example, there can be one set $l'_1 \dots l'_i$ representing the executed actions $\{a_1 a_2\}$ and a second set $l'_{i+1} \dots l'_n$ representing the executed actions $\{a_2 a_3\}$. These sets can be *overlapping*. This means, that in the task instance list, an n-gram of one set overlaps with an n-gram of another set. For example, the task instance list may contain a sublist of action instances $\{a'_1 a'_2 a'_3\}$. The first and the second action instance in this sublist belong to the n-gram set $l'_1 \dots l'_i$, as they represent the executed actions $\{a_1 a_2\}$. The second and the third action instance in this sublist belong to the n-gram set $l'_{i+1} \dots l'_n$, as they represent an execution of $\{a_2 a_3\}$. In this case, we need to decide, which of both n-gram sets needs to be handled first. If $l'_1 \dots l'_i$ is handled first, $\{a'_1 a'_2\}$ would be replaced by a sequence instance while an n-gram in $l'_{i+1} \dots l'_n$ gets destroyed. If $l'_{i+1} \dots l'_n$ is handled first, the situation is the other way around.

To decide, which of the n-gram sets needs to be replaced first, the sequence detection compares the n-grams sets with each other. It determines those sets that contain the maximum number of n-grams and discards all other sets. This ensures to handle those n-gram sets first that represent action and task combinations being performed most often by users and being, hence, more representative for a task execution than others. From the remaining sets, it determines those whose n-gram are the longest, i.e., have the largest n, and discards all other sets. This ensures that longer action and task combinations are preferred instead of shorter ones as they represent more complex tasks.

After this initial choosing, there can still be overlapping n-gram sets. For example, there may remain the two n-gram sets $l'_1 \dots l'_i$ and $l'_{i+1} \dots l'_n$ mentioned above. This happens if they contain the same number of n-grams and because the n-grams in both sets have the same length. Therefore, we perform a further choosing to decide, which of the n-gram sets needs to be handled first. For this, we first check if the remaining sets are indeed overlapping, i.e., if there is a sublist $\{a'_1 a'_2 a'_3\}$ in the task instance list. If not, we handle the remaining sets because the contained n-grams can be replaced independent from each other.

If there are overlapping sets, we perform a further choosing of the n-gram sets to be handled. For this, we iteratively discard n-gram sets causing overlaps. For this process, we introduce the two terms *succeeding overlap* and *preceding overlap*. An n-gram l'_i has a succeeding overlap to an n-gram l'_j if both n-grams overlap and if l'_i starts later in the task instance list than l'_j. In this situation, l'_j has a preceding overlap with l'_i as it starts earlier in the task instance list. Using this terminology, the discarding process works as follows:

1. determine all overlapping n-gram sets
2. if there is exactly one n-gram set causing most overlaps, discard it and go back to 1.
3. if there is exactly one n-gram set causing most succeeding overlaps, discard it and go back to 1.
4. if there is exactly one n-gram set causing least preceding overlaps, discard it and go back to 1.
5. discard the n-gram set containing the n-gram being the last succeeding overlap in the task instance list and go back to 1.

If at some point in this discarding process, no further overlapping n-gram sets exist, the process is finished and the remaining n-gram sets are handled. The process always terminates as at some point the last succeeding overlap is discarded and, through this, no further overlaps can exist. Through its nature, the process detects sequences in a predictive order and does create random structures. The discarded n-gram sets are handled in a subsequent sequence detection. This takes place due to the alternating iteration and sequence detection, which we describe in the following subsection.

4.4.2.3. Alternating Iteration and Sequence Detection

As shown in Figure 4.5, the iteration and sequence detection are repeated alternately. If the iteration detection is applied after a sequence detection, it identifies subsequent instances of the same sequence. As with action instances, these are replaced with instances of an iteration having the corresponding sequence as its single child. In addition, the sequence detection can detect and replace identical n-grams containing other sequences. Through this, at any iteration and sequence detection cycle, the task instance list becomes shorter and consists of more and more sequence and iteration instances. Additionally, the task trees become more complex and deeper. As shown in Figure 4.5, the process stops, when no further sequences or iterations are detected. In the end, the task instance list contains instances of detected iterations and sequences, as well as action instances. These action instances are noise in the data or belong to tasks that were executed only once. These tasks are not detected as our approach requires at least two executions of a task to be detectable.

Within a single task instance list, users may perform a specific task only once. For example, in a single user session, users usually perform a login process only once. If our approach is applied only on one task instance list, these tasks are not found as our approach requires, that tasks are executed at least twice to be detected. Therefore, we apply the iteration and sequence detection not on a single task instance list, but on several task instance lists at once. This means, the approach is applied on recordings of several users and several user sessions. Through this, we ensure that those tasks being executed seldom in a single user session but occur in many user sessions are more likely to be detected. This is important as the generated task trees intend to represent actual user behavior.

When being applied on several task instance lists, our process for choosing the n-gram sets to be handled first needs to be adapted. This is required, because the last rule in the process considers only one task instance list, so far. Hence, we adapt the rules to be applicable on several task instance lists. The basis for this adaptation are the positions of n-grams in task instance lists. Each n-gram l'_i in an n-gram set $l'_1 \dots l'_n$ has a specific position $p(l'_i)$ in a task instance list. An example is an n-gram set $l'_1 \dots l'_n$ representing the executed actions $\{a_2 a_3\}$. If there is a task instance list $\{a'_1 a'_2 a'_3\}$, then there is an n-gram $l'_i = \{a'_2 a'_3\} \in l'_1 \dots l'_n$ which represents the second and the third element in this task instance list. The position $p(l'_i)$ in this task instance list is 2. In this manner, we determine positions for any n-gram in an n-gram set $l'_1 \dots l'_n$. Based on the individual positions, we define a metric for all n-grams in $l'_i \in l'_1 \dots l'_n$. This metric is the sum of all positions of all n-grams in an n-gram set, i.e., $\sum_{l'_1 \dots l'_n} p(l'_i)$.

Based on the position information for n-grams and the above metric, we adapt our process for choosing n-gram sets to be handled first. We replace the last rule by two other rules, which take the position information into account. The first of the new rules considers the sum of the positions of all n-grams in an n-gram set. The second focuses only on the smallest position of any n-gram in an n-gram set. The extended choosing process is as follows:

1. determine all overlapping n-gram sets
2. if there is exactly one n-gram set causing most overlaps, discard it and go back to 1.
3. if there is exactly one n-gram set causing most succeeding overlaps, discard it and go back to 1.
4. if there is exactly one n-gram set causing least preceding overlaps, discard it and go back to 1.
5. determine the n-gram sets for which $\sum_{l'_1 \dots l'_n} p(l'_i)$ is minimized and, if these are not all detected n-gram sets, discard all other n-gram sets
6. determine the n-gram sets containing an n-gram l'_i for which $p(l'_i)$ is minimized and, if these are not all detected n-gram sets, discard all other n-gram sets

Also this extended process finishes at any position if there are no further overlaps. Then the remaining n-gram sets are handled. This process might not terminate anymore. The reason is, that even the last two rules may not be distinctive enough and several n-gram sets might match all criteria in the same way. For example, there can be an n-gram set representing the actions $\{a_1a_2\}$ and a further n-gram set representing the actions $\{a_2a_1\}$. If there are two task instance lists, one starting with $\{a'_1a'_2a'_1\}$ and the other starting with $\{a'_2a'_1a'_2\}$, then there are exactly two n-grams in both n-gram sets. Both n-gram sets contain an n-gram l'_1 having the smallest position $p(l'_i) = 0$ in the task instance lists. In this situation, we cannot decide anymore, which of the n-gram sets needs to be handled first. Therefore, we throw an exception and terminate the whole task tree generation at this point. We consider this a fail of the task tree generation. In our case studies, we evaluated how often the task tree generation fails and how this influences the task tree generation process.

4.4.3. Merging of Similar Sequences

The task trees generated through iteration and sequence detection consist only of sequences and iterations [20]. They cannot describe execution variants like optional actions or a choice between subtasks. Hence, for each execution variant performed by users, the approach generates separate task trees. This increases the number of generated tasks and makes a subsequent usability evaluation more challenging. For example, users usually carry out two variants of performing a login process on a website. These are distinct in the navigation between the user name field and the password field [21]. Some users use a mouse click on the password field, others the tabulator key for this navigation. For these variants, the iteration and sequence detection generates two similar task trees, one containing the mouse click navigation, the other the tabulator key navigation. Nonetheless, both task trees semantically describe the same task, i.e., performing a login. To handle execution variants, subsequent to the iteration and sequence detection we perform a detection and merging of similar sequences as shown in Figure 4.4. We focus on sequences, as merging iterations mainly consists of merging their children, which are either actions or sequences.

Our approach for detecting and merging similar sequences consists of several substeps which are shown in Figure 4.7. The substeps are: detection of similar sequences, choosing of sequences to merge, adaptation of flattened instances, iteration detection, sequence detection, and harmonization of parent tasks. The substeps are repeated until no further similar sequences are detected. The details of the substeps are described in the following subsections.

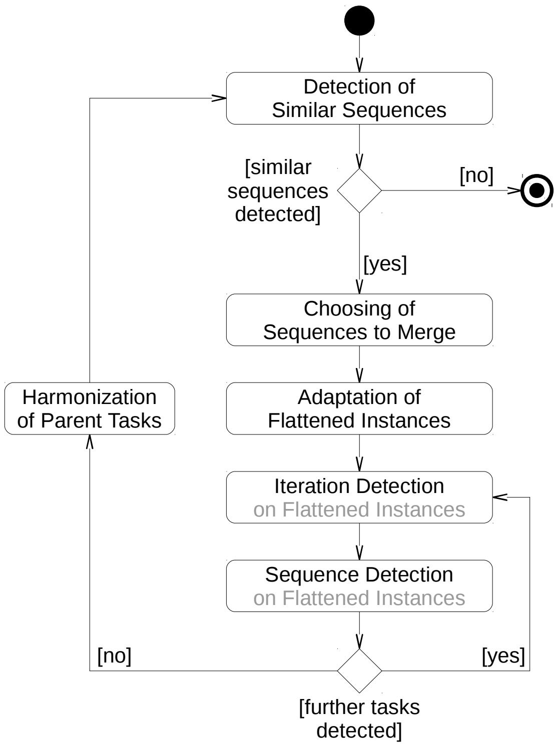

Detection of Similar Sequences
[similar sequences detected]
[no]
[yes]
Choosing of Sequences to Merge
Harmonization of Parent Tasks
Adaptation of Flattened Instances
Iteration Detection on Flattened Instances
Sequence Detection on Flattened Instances
[no]
[yes]
[further tasks detected]

Figure 4.7.: Process for detection and merging of similar sequences (adapted from [21]).

4.4.3.1. Detection of Similar Sequences

Detecting similar sequences is the first substep of merging similar sequences (see Figure 4.7). For this we use a similarity measure for two sequences [21]. We call this measure *sequence similarity*. We refer to the sequence similarity for two sequences s_1 and s_2 as $sim(s_1, s_2)$.

The sequence similarity is based on the leaf nodes, i.e., actions, of the task tree belonging to a sequence. We start by creating a list $L(s)$ of these actions for each sequence s, which we call *task list*. The actions in the task list are in the order in which, they would be executed if the sequence was performed with any iteration in the task tree repeated exactly once. Figure 4.8 shows an example for a full merging process for two similar sequences. There, Figure 4.8 a displays the task trees for two sequences s_1 and s_2. For simplification, the actions are named with single characters. Figure 4.8 b shows $L(s_1)$ and $L(s_2)$ containing the actions of s_1 and s_2 in the order of minimal execution.

After the creation of the task lists, we compare them with each other. The comparison is done using a diff algorithm similar to those used for the comparison of texts. In our work, we use Myers diff algorithm [89] designed for comparing texts, which we adapted to be used for the comparison of task lists, instead. The result of the comparison of two lists $L(s_1)$ and $L(s_2)$ is a list of deltas $d_1 \dots d_n$ of three different types:

- *insert*: an action contained in $L(s_2)$ at a specific position is not contained in $L(s_1)$ at the corresponding position
- *delete*: an action contained in $L(s_1)$ at a specific position is not contained in $L(s_2)$ at the corresponding position
- *change*: an action contained in $L(s_1)$ at a specific position is replaced by another action in $L(s_2)$ at the corresponding position

The number of actions covered by a delta d_i in both lists is $|d_i|$. An example for deltas identified for the two lists $L(s_1)$ and $L(s_2)$ in Figure 4.8 b is also shown in the figure. The deltas are an insert of Action b at the second position, a delete of Action d at the third position and a change of Action f to Action g at the fifth position, where positions are counted for $L(s_1)$.

For the list of deltas between two task lists, we determine the number of all actions in both lists $L(s_1)$ and $L(s_2)$ that belong to a delta, i.e., $\sum_{i=1}^{n} |d_i|$. For calculating the sequence similarity $sim(s_1, s_2)$, we divide this sum by the number of all actions belonging to both lists $L(s_1)$ and $L(s_2)$ and subtract the result from 1. Hence, the sequence similarity is calculated as follows:

$$sim(s_1, s_2) = 1 - \frac{\sum_{i=1}^{n} |d_i|}{|L(s_1)| + |L(s_2)|} \tag{4.4.1}$$

The utilized diff algorithm is not necessarily commutative. This means, the list of deltas returned by the diff algorithm may differ if the left and the right hand side of the comparison are swapped. Therefore, the sequence similarity as calculated above would depend on the order in which $L(s_1)$ and $L(s_2)$ are given to the diff algorithm. To ensure that we always get the same similarity for two sequences s_1 and s_2, we apply the diff algorithm twice, i.e.,

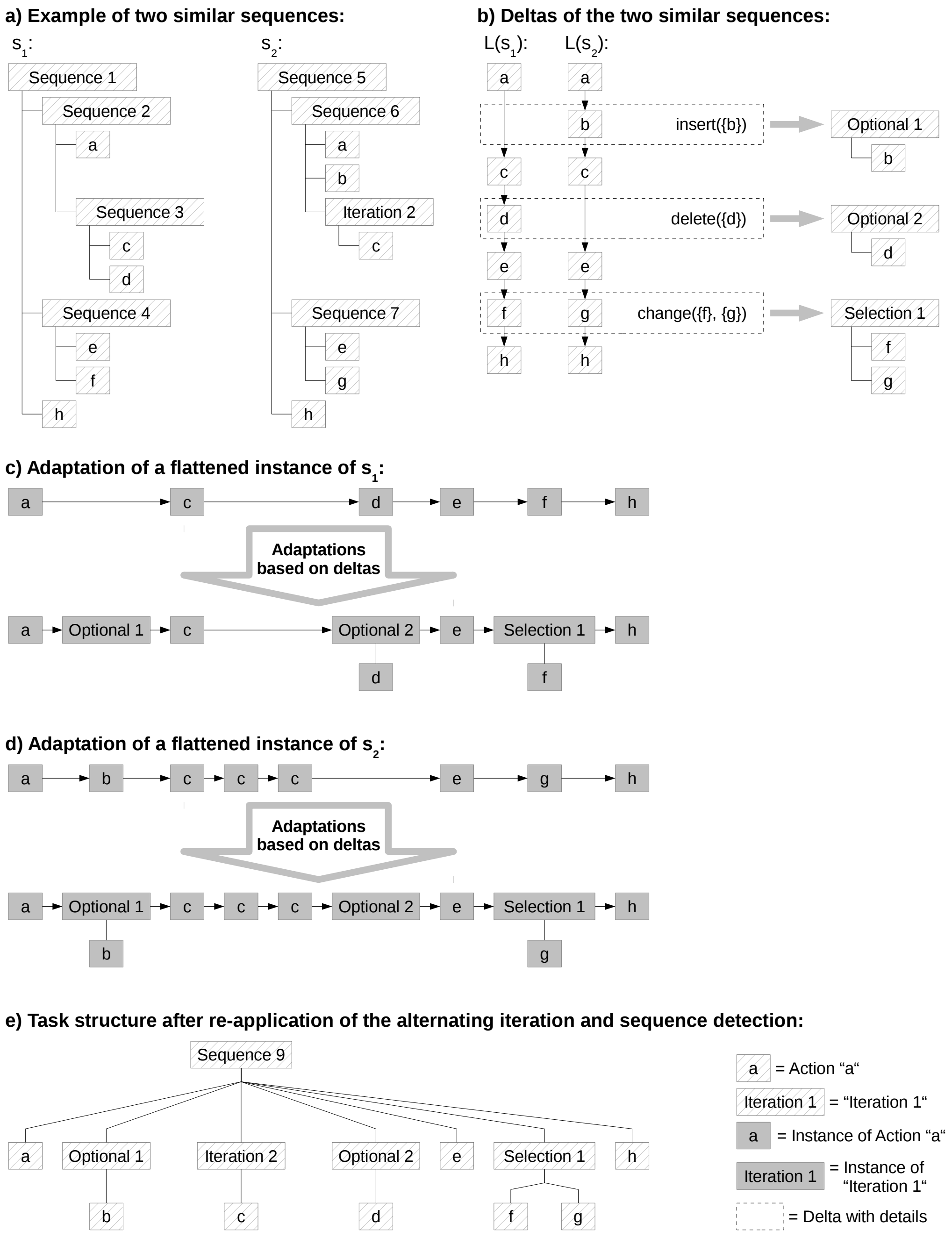

a) Example of two similar sequences:
s1:
Sequence 1
Sequence 2
a
Sequence 3
c
d
Sequence 4
e
f
h
s2:
Sequence 5
Sequence 6
a
b
Iteration 2
c
Sequence 7
e
g
h
b) Deltas of the two similar sequences:
L(s1):
L(s2):
insert({b})
delete({d})
change({f}, {g})
Optional 1
Optional 2
Selection 1
c) Adaptation of a flattened instance of s1:
Adaptations based on deltas
d) Adaptation of a flattened instance of s2:
e) Task structure after re-application of the alternating iteration and sequence detection:
Sequence 9
a = Action "a"
Iteration 1 = "Iteration 1"
a = Instance of Action "a"
Iteration 1 = Instance of "Iteration 1"
= Delta with details

Figure 4.8.: Example for merging two similar sequences (adapted from [21]).

first with $L(s_1)$ and second with $L(s_2)$ as the left hand side of the comparison. Then, we calculate the sequence similarity for both lists of deltas. The resulting sequence similarity for s_1 and s_2 is then the maximum of the two individually calculated sequence similarities. Furthermore, the subsequent merging process considers only the deltas that cause the higher sequence similarity.

The sequence similarity $sim(s_1, s_2)$ is 1 for sequences sharing the same task lists. But as it does not consider the structure of the sequences task trees, the sequences may still be different. The similarity of two sequences will be lower the more deltas are contained in their task lists. At a minimum, $sim(s_1, s_2)$ can be 0, indicating no common actions in the task lists and, hence, no similarity between the sequences.

To detect similar sequences, we compare all sequences resulting from the iteration and sequence detection with each other and calculate their similarities. This results in a list of sequence pairs with an attributed similarity. In the next substep, we choose from this list those pairs that need to be merged.

4.4.3.2. Choosing Sequences to Merge

The list of sequence pairs and their similarities resulting from the previous substep contains all pairs of sequences determined in the iteration and sequence detection. This also includes pairs for which a merging is not reasonable. For example, there might be pairs with a rather low similarity. Therefore, we perform a choosing of sequence pairs to be merged as the second substep for merging similar sequences (see Figure 4.7). For this, we start by choosing only those pairs $(s_i, s_j) | i \neq j$

- whose similarity $sim(s_i, s_j)$ exceeds a certain threshold called sim_{min},
- for which $sim(s_i, s_j)$ is maximal and which are, therefore, the most similar ones,
- whose deltas are covering neither the first nor the last elements of the task lists $L(s_i)$ and $L(s_j)$, and
- for which neither s_i nor s_j is a direct or indirect parent of any sequence of another pair.

Through these conditions, we ensure several aspects. In the first place, sequences are only considered similar if at least a minimum of similarity, i.e., sim_{min}, is reached. Secondly, with the third condition we ensure that we always consider entire similar sequences. If we, instead, considered similarities, where the deltas are at the beginning or end of the task lists, there might be a parent sequence of either sequence having a higher similarity to the respective other sequence of the pair. Finally, the resulting pairs will all be independent from each other, in reference to the parent child relationships between tasks. Hence, subsequently we do not merge a sequence with either one of its parents or one of its children in the same merging cycle.

The list resulting from this initial choosing contains only sequence pairs with the same similarity. Several of these pairs might refer to the same sequence. For example, a sequence s_1 may represent a default execution of a users task. At the same time, two further sequences s_2 and s_3 may represent two different variants of s_1. The sequence similarities of such an example can be equal, i.e., $sim(s_1, s_2) = sim(s_1, s_3)$. In such a case, we need to decide, which sequence pair sharing the same sequence needs to be merged first. For this, we initially determine the set S containing all sequences shared by more than one pair. If $S = \emptyset$, there are no conflicts and all pairs will be merged. If $S \neq \emptyset$, we drop those pairs not referring to a sequence $s \in S$ and perform a choosing between the remaining pairs. This choosing is done by sorting and filtering the remaining pairs. This results in a pair list that starts with the pair to be merged first and ends with the pair to be merged last. The sorting is done based on the following rules for comparing two pairs. The rules are applied in their given order, meaning that if a rule is not able to make a distinction between the pairs, then the next rule is tried. The rules are:

- the pair whose number of action instances from which both sequences are generated, i.e., $|a'(s_1)| + |a'(s_2)|$, is higher, precedes in the list
- the pair whose number of instances of both sequences, i.e., $|x(s_1)| + |x(s_2)|$, is higher, precedes in the list
- the pair whose sum of the depths of both sequences, i.e., $depth(s_1) + depth(s_2)$, is higher, precedes in the list
- the pair whose sum of the instances of both sequences and all their direct and indirect subtasks is higher, precedes in the list

The last rule is based on the fact, that any child of a sequence s can also be a child of another task and its number of instances might be higher than $|x(s)|$. After its sorting, we perform a filtering of the list. We drop those pairs that refer to an $s \in S$ which is already referred by another pair preceding in the list. As an example, we consider $\{s_1, s_2\}, \{s_1, s_3\}, \{s_1, s_4\}$ as a sorted list of sequence pairs. All pairs in this list refer to s_1. The filtering discards the second and the third pair from the list, because the first pair already references s_1. Hence, the subsequent pairs in the list also referencing s_1 need to be discarded.

There may be pairs for which the above rules cannot provide a unique sorting. This happens if the compared values for two pairs are the same. In this situation, the sorting process puts the pairs next to each other in the list without a predefined order. The subsequent filtering then ensures that still always the same sequence pairs will be discarded. For this, it performs a check before discarding a pair $\{s_i, s_j\}$ from the list. In this check, it ensures that the above rules are able to create a sorting for $\{s_i, s_j\}$ and the preceding pairs in the list referring to s_i or s_j. If the rules cannot provide a sorting, we throw an exception and terminate the process as the filtering cannot decide, which of the pairs should remain in list. In the above example, the filtering terminates if the rules cannot provide a sorting for the first and the second, or for the first and the third element in the list. If the filtering terminates in this way, we consider the sequence merging as failed. In our case studies, we evaluate how often the sequence merging fails and how this influences the sequence merging process.

After the sorting and the filtering, the list contains only pairs referring to different sequences. Hence, we perform the merging of the remaining pairs in the order given by the list. The full choosing process for sequence pairs is shown as pseudocode in Algorithm 4.2. There, the Lines 4 to 7 show the initial choosing of pairs that show a minimum similarity and that are not parents of sequences of other pairs in *P*. Lines 9 and 11 check if there are multiple remaining pairs referring to the same sequence. If so, Line 12 drops all pairs not referring to the same sequences from *P*. In Lines 14 and 15, we do the initial sorting of the remaining pairs, which is stored in the sorted set P'. Then, in Lines 17 to 26, we perform the filtering. Line 20 checks if the filtering can decide which pair should be discarded. As a result, the sorted set *P* contains the remaining pairs which need to be merged.

Algorithm 4.2 Choosing of sequence pairs to be merged.

```
1:  $P \leftarrow \{$sequence pairs and their similarity level$\}$
2:  $sim_{max} \leftarrow \max\{$similarity level of pairs in $P\}$
3:
4:  $P \leftarrow P \setminus \{(s_i, s_j) \mid sim(s_i, s_j) < sim_{min}\}$
5:  $P \leftarrow P \setminus \{(s_i, s_j) \mid$ deltas are at beginning or end of $L(s_1)$ or $L(s_2)\}$
6:  $P \leftarrow P \setminus \{(s_i, s_j) \mid sim(s_i, s_j) < sim_{max}\}$
7:  $P \leftarrow P \setminus \{(s_i, s_j) \mid s_i$ or $s_j$ are parent of any other sequence referred by another pair$\}$
8:
9:  $S \leftarrow \{$sequences belonging to several pairs in $P\}$
10:
11: if $S \neq \emptyset$ then
12:     $P \leftarrow P \setminus \{(s_i, s_j) \mid s_i, s_j \notin S\}$
13:
14:     $C \leftarrow$ comparator        // for applying sort rules
15:     $P' \leftarrow$ sort($P$ using $C$)
16:
17:     $P \leftarrow \emptyset$
18:     for all $(s_i, s_j) \in P'$ do
19:         if $\exists (s'_i, s'_j) \in P \mid (s_i = s'_i \vee s_i = s'_j \vee s_j = s'_i \vee s_j = s'_j)$ then
20:             if $C$ cannot provide a sorting for $(s_i, s_j)$ and $(s'_i, s'_j)$ then
21:                 // throw Exception
22:             end if
23:         else
24:             $P \leftarrow P \cup \{(s_i, s_j)\}$
25:         end if
26:     end for
27: end if
28:
29: // merge remaining pairs in $P$
```

4.4.3.3. Flattening and Adaptation of Sequence Instances

After having detected and chosen the sequences to be merged, we perform the next substep for merging similar sequences, which is the flattening and adaptation of sequence instances (see Figure 4.7). We start by considering flattened instances of the sequences to be merged. A flattened instance of a task instance t' is an ordered task instance list $L'(t')$ of the action instances, i.e., leaf nodes, of t' in the order they were performed by the user. This is similar to a task list $L(s)$ of a sequence s, except that it contains action instances instead of actions[2]. Examples of flattened task instances for the tasks in Figure 4.8 a are shown in the upper lists in Figure 4.8 c and Figure 4.8 d. The lists consist of grey boxes representing action instances. The corresponding actions are identified through letters in the boxes. The arrows between the boxes denote the order of execution of the action instances. We create flattened instances out of all instances of two sequences to be merged.

Afterwards, for two sequences s_1 and s_2 to be merged, we consider the deltas between their task lists $L(s_1)$ and $L(s_2)$, which were identified in the previous substep. For each insert or delete delta, we generate a new task of type optional. The single child of this optional is the action that is either inserted or deleted. For each change delta, we generate a new task of type selection. This selection gets both execution variants denoted by the delta as its new children. In Figure 4.8 b, we show examples for generated optionals and selections for a list of deltas. In this figure, *Optional 1* is generated for the insert delta of Action b and, therefore, has the action as its child. *Selection 1* is generated for the change delta of the actions f and g and, thus, has both actions as its children.

Using the optionals and selections generated for the deltas, we adapt the flattened instances of two sequences s_1 and s_2 to be merged. For this, we integrate instances of the optionals and selections into the flattened instances. This integration is done based on the deltas and the knowledge where these deltas are located in the task lists $L(s_1)$ and $L(s_2)$.

For each insert or delete delta, we first consider the sequence where the actions denoted by the delta are missing. In each flattened instance of this sequence, we integrate an instance of the optional created for the delta. The position of the optional instance depends on the delta and corresponds to the position where the corresponding action was left out in the execution. The optional instances have no children to indicate, that the execution of the action was left out. We then consider the other sequence of the pair. In each flattened instance of this sequence, we replace the instance of the action denoted by the delta with an instance of the optional created for the delta. This optional instance gets the replaced action instance as child to denote that the action was performed. An example for a replacement of action instances for insert and delete deltas is shown in Figure 4.8 c and Figure 4.8 d. There, the insert delta of Action b is handled through integrating instances of *Optional 1* into the flattened instances of s_1 and s_2. In the flattened instance of s_1 (Figure 4.8 c), an instance

[2]$L'(t')$ is also similar to $a'(t')$ but only in its simplest form. In adaptations of our approach described in Section 4.4.3.7 and Section 4.4.3.8, $L'(t')$ contains other elements than action instances and is, hence, different from $a'(t')$.

of *Optional 1* without a child is integrated to indicate, that the execution of the optional Action b was left out. In contrast, in the flattened instance of s_2 (Figure 4.8 d), an instance of *Optional 1* replaces the instance of Action b and gets it as its child. This indicates, that here Action b was performed although it could have been left out.

The handling of change deltas is similar to that of insert and delete deltas. Instead of optional instances, we create instances of the selection representing the delta. These instances replace the instances of the actions denoted by the delta. An example of the handling of a change delta is also shown in Figure 4.8 c and Figure 4.8 d. The instances of actions f (Figure 4.8 c) and g (Figure 4.8 d) are replaced by instances of *Selection 1*, which represents the corresponding delta. The selection instance gets the replaced action instances as children.

A single delta may cover several actions at once in $L(s_1)$ and/or $L(s_2)$. In this case, we create optionals and selections that have intermediate sequences as their children. These sequences in turn have the actions belonging to the delta as their children. For example, if an insert delta denotes an insert of the actions $a_i \ldots a_j$, then the single child of the corresponding optional o is a sequence s having $a_i \ldots a_j$ as its children. The same applies for change deltas. Introducing these intermediate sequences also has an effect on the adaptation of the flattened instances. If an intermediate sequence is required for a delta, then the adaptation of a flattened instance, which is done to reflect the delta, also creates instances of the intermediate sequence. For example, for the above mentioned insert delta, the flattened instances of one of the merged sequences contains executions of $a_i \ldots a_j$. These need to be replaced by an instance of the optional o. This optional instance gets a single child, which is an instance of the intermediate sequence s. This sequence instance in turn gets the replaced instances of $a_i \ldots a_j$ as its children.

After all adaptations are done, the flattened instances of the two sequences to be merged consist of action, optional, and selection instances. Their ordering corresponds to a common task model, which will be determined in the next substeps for merging similar sequences.

4.4.3.4. Iteration and Sequence Detection on Adapted Flattened Sequence Instances

On the adapted flattened instances of two sequences to be merged, we apply an iteration and sequence detection as described in Section 4.4.2. This is also done alternately as shown in Figure 4.7 until no further iterations and sequences are detected. For this, the flattened instances become task instance lists, which in turn become the input of the iteration and sequence detection. Due to the adaptations, the flattened instances have similar orders and share common n-grams of action, optional, and selection instances. Furthermore, the iteration and sequence detection is applied on at least two flattened instances, one for each sequence to be merged. Therefore, at the end of its application, the iteration and sequence detection always finds a single sequence. This single sequence is the result of the merge. Figure 4.8 e shows the sequence resulting from the iteration and sequence detection on the adapted flattened instances in Figure 4.8 c and Figure 4.8 d, which is the result of merging the sequences in Figure 4.8 a.

4.4.3.5. Harmonization of Parent Tasks

After we calculated a result s_3 for the merging of two sequences s_1 and s_2, we need to update the parent tasks that have s_1 or s_2 as any of their children. We do this in the next substep of our sequence merging process shown in Figure 4.7. This substep covers the replacement of s_1 and s_2 with s_3 in any child list of any other task. This adaptation is done by a simple child replacement. Examples for updates of two parent tasks, after their children have been merged, are shown in Figure 4.9. Figure 4.9 a shows two sequences *Sequence 2* and *Sequence 3* (indicated through dotted boxes) which are similar and, hence, merged to become *Sequence 4* in Figure 4.9 b. Through this, *Iteration 1* and *Iteration 2*, which are the parent tasks of *Sequence 2* and *Sequence 3*, are also updated having now the new child *Sequence 4*.

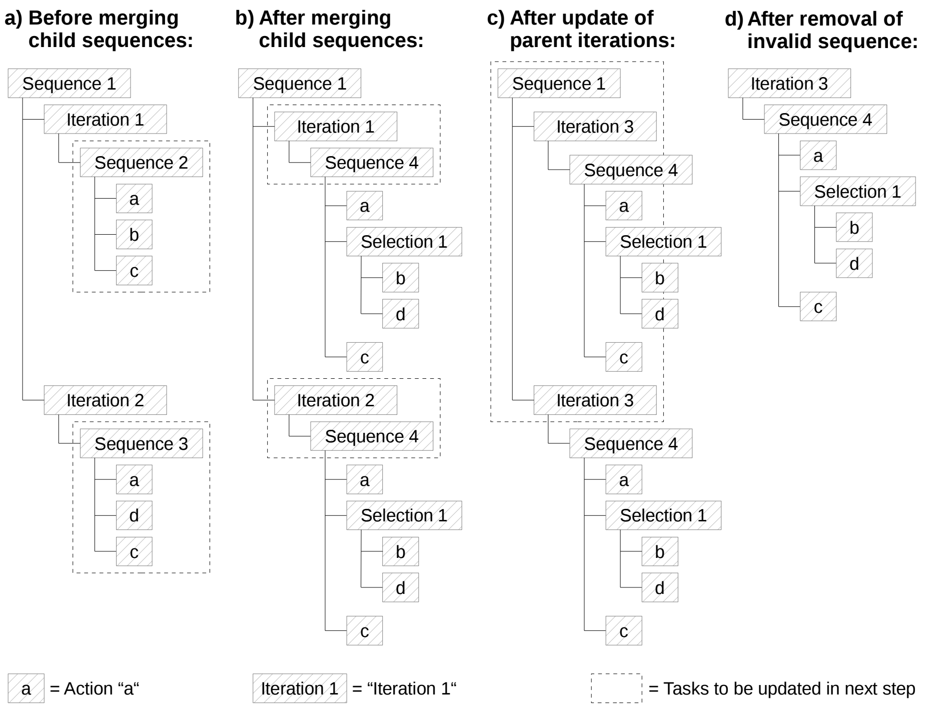


Figure 4.9.: Example for updating parent tasks after merging two similar sequences.

In addition to the update of the parent tasks, we also adapt their instances. This is required to ensure, that the instances match the corresponding model. The adaptation is done based on the flattened instances of s_1 and s_2, which became task instance lists and also input for the iteration and sequence detection. After the iteration and sequence detection, these task instance lists all have a length of one and the sole elements in the lists are instances of s_3. These instances are the replacements for the instances of s_1 or s_2. As we know, which instance of s_1 and s_2 was flattened and transformed into a specific task instance list, we also know which instance of s_3 needs to replace the corresponding instance of s_1 or s_2. We use this traceability to perform the respective replacements in the instances of the parent tasks.

Due to the replacements, parent tasks can become unharmonized. For example, there may be two parent iterations i_1 and i_2, where the single child of i_1 is s_1 and the single child of i_2 is s_2. After a merge of s_1 and s_2, the children of i_1 and i_2 are replaced by the merge result s_3. Due to this, i_1 and i_2 become identical. To ensure a homogeneous task tree, we also replace i_1 and i_2 with a new iteration i_3, which has s_3 as its single child. This also implies an adaptation of the corresponding instances. An example for this situation is shown in Figure 4.9 b and 4.9 c. There, after the merging of their children, *Iteration 1* and *Iteration 2* have the same child. Hence, we replace both of them with *Iteration 3*, which has the merge result *Sequence 4* as its single child.

By merging sequences, but also by harmonizing iterations, two subsequent children of a parent sequence may become identical. This situation is shown in Figure 4.9 c, which resulted from a harmonization of iterations. There *Sequence 1* has two subsequent children, which are the same iteration, namely *Iteration 3*. In our task trees, this must be represented as one iteration. Hence, we detect these situations and replace two subsequent identical children with a single iteration of them. If the children are already iterations, as in the example, the iteration is reused. This update is done correspondingly on the instances of the affected parent tasks, as well. In addition, if a sequence only has two children, which become identical, the whole sequence is discarded and all its occurrences are replaced by an iteration of the identical children. This situation is shown in Figure 4.9 c and 4.9 d. Through the merging of their children, the child iterations of *Sequence 1* become identical, i.e., *Iteration 3*. Hence, any occurrence of *Sequence 1* will be replaced by an iteration of its identical children. In this example, these children are already iterations, namely *Iteration 3*. Therefore, *Iteration 3* is used for the replacement of *Sequence 1*.

Through the harmonization of parent tasks, grandparent tasks may also require a harmonization. The corresponding adaptations are identical to those mentioned for the parent tasks. We perform them until no further parent tasks require an adaptation.

4.4.3.6. Repetition of Sequence Merging

As shown in Figure 4.7, the merging of similar sequences is repeated multiple times. This is required, as during a single merge only some pairs of similar sequences are chosen to be merged. The merging is repeated until no further pairs of similar sequences are chosen to be merged. This may happen, e.g., if the similarity level of the pairs does not exceed sim_{min}.

During several mergings, identical optionals, selections, or intermediate sequences may be detected. An example are optionals that have the same action as their single child. For all mergings, we ensure, that previously created optionals, selections, and intermediate sequences are reused. It may also be the case, that the result of a merge is a sequence having the same children as a previously detected intermediate sequence. In this case, we also reuse the intermediate sequence to not create a new one. Through this, the task trees remain harmonized.

In a merging cycle, a sequence pair to be merged might contain the result of a previous merge. This includes optionals, selections, and new intermediate sequences. Such sequence pairs require a specific handling in the merging process. This, and other exceptional handlings during the merge process, are described in the following subsections.

4.4.3.7. Adaptations of the Sequence Similarity Calculation

The sequence similarity as described in Section 4.4.3.1 does not yet consider exceptional situations. These situations as well as their consideration in the calculation of $sim(s_1,s_2)$ are described in this section. One situation is, that during task execution, users perform inefficient actions like scrolling, which do not contribute semantically to the task. Due to the iteration and sequence detection, as described in Section 4.4.2, these actions are part of the task trees generated so far. Hence, they may also occur in the task lists, which are the basis for calculating $sim(s_1,s_2)$. When comparing two task lists, a high number of inefficient actions in both lists may cause a high similarity of these lists. But from a semantic point of view, the lists can be very different. Hence, we adapt the calculation of the sequence similarity to be higher for two sequences, which have common efficient actions, and respectively lower for two sequences, which share inefficient actions. To achieve this, we consider inefficient actions also as deltas between two lists. Let n_{ineff} be the number of inefficient actions in two task lists $L(s_1)$ and $L(s_2)$, which are not part of any delta between $L(s_1)$ and $L(s_2)$. Then we adapt the calculation of the sequence similarity $sim(s_1,s_2)$ for the two sequences s_1 and s_2 as follows:

$$sim(s_1,s_2) = 1 - \frac{n_{ineff} + \sum_{i=1}^{n} |d_i|}{|L(s_1)| + |L(s_2)|} \tag{4.4.2}$$

Through this adaptation, $sim(s_1,s_2)$ cannot become 1, even if $L(s_1) = L(s_2)$, as long as $L(s_1)$ contains an inefficient action. In addition, it contradicts the idea of the sequence similarity. Hence, we adapt the calculation of the sequence similarity further to be 1 if $L(s_1) = L(s_2)$ and as previously defined if $L(s_1) \neq L(s_2)$:

$$sim(s_1,s_2) = \begin{cases} 1 - \frac{n_{ineff} + \sum_{i=1}^{n} |d_i|}{|L(s_1)| + |L(s_2)|} & L(s_1) \neq L(s_2) \\ 1 & L(s_1) = L(s_2) \end{cases} \tag{4.4.3}$$

A further exceptional situation results from the repetition of merging similar sequences. In merging cycles following the first merge, the pairs of similar sequences may include results of a previous merge. This means, that the compared sequences may have direct or indirect children, which are optionals or selections. When calculating the task lists for two sequences, optionals do not need a specific handling. The actions covered by an optional are simply included in the task lists. On the contrary, for selections, it is not clear, in which order the actions covered by their children have to be included in the task lists. Therefore, we do not include the actions denoted by a selection, but the selection itself in the task lists.

The calculation of $sim(s_1, s_2)$ so far considers only actions in the task lists. But a selection may cover multiple actions at once and, therefore, represents multiple actions at once as a single entry in a task list. Through this, the number of actions, represented by entries in the task lists, becomes unbalanced. We consider this in a further adaptation of the calculation of the sequence similarity. For this, we introduce a function $a(t)$ that returns the average number of actions covered by a task t. This function is defined recursively and its result depends on the children and the type of t. The function returns the following:

$$a(t) = \begin{cases} \sum_{c_i \in c(t)} a(c_i) & \text{if } t \text{ is a sequence} \\ a(c_1) & \text{if } t \text{ is an iteration or optional and } c_1 \text{ its single child} \\ \frac{\sum_{c_i \in c(t)} a(c_i)}{|c(t)|} & \text{if } t \text{ is a selection} \end{cases} \tag{4.4.4}$$

If a child c_i of a task is an action, then $a(c_i) = 1$. In addition, we define the function $a(L)$ also for a task list L. Here, it returns the sum of the average number of actions covered by the tasks $t \in L$ plus the number of actions $a \in L$. This means, $a(L)$ is calculated as follows:

$$a(L) = \sum_{t \in L} a(t) + |\{a \mid a \in L\}| \tag{4.4.5}$$

Correspondingly, for a delta d, we define the function $a(d)$. This returns the sum of the average number of actions covered by the tasks $t \in d$ plus the number of actions $a \in d$. This means, $a(d)$ is calculated as follows:

$$a(d) = \sum_{t \in d} a(t) + |\{a \mid a \in d\}| \tag{4.4.6}$$

We use $a(L)$ and $a(d)$ to adapt the calculation of the sequence similarity as follows:

$$sim(s_1, s_2) = \begin{cases} 1 - \frac{n_{ineff} + \sum_{i=1}^{n} a(d_i)}{a(L(s_1)) + a(L(s_2))} & L(s_1) \neq L(s_2) \\ 1 & L(s_1) = L(s_2) \end{cases} \tag{4.4.7}$$

Through this adaptation, we ensure that selections in task lists get a weight corresponding to the average number of actions they represent.

4.4.3.8. Adaptations of the Sequence Flattening

The above approach for merging similar sequences discards common substructures of both sequences. For example, if the first child of two similar sequences s_1 and s_2 is the same task (as shown in Figure 4.10 a), then instances of this task would be discarded by flattening the instances $x(s_1)$ and $x(s_2)$. In addition, they would not be found again on the subsequent

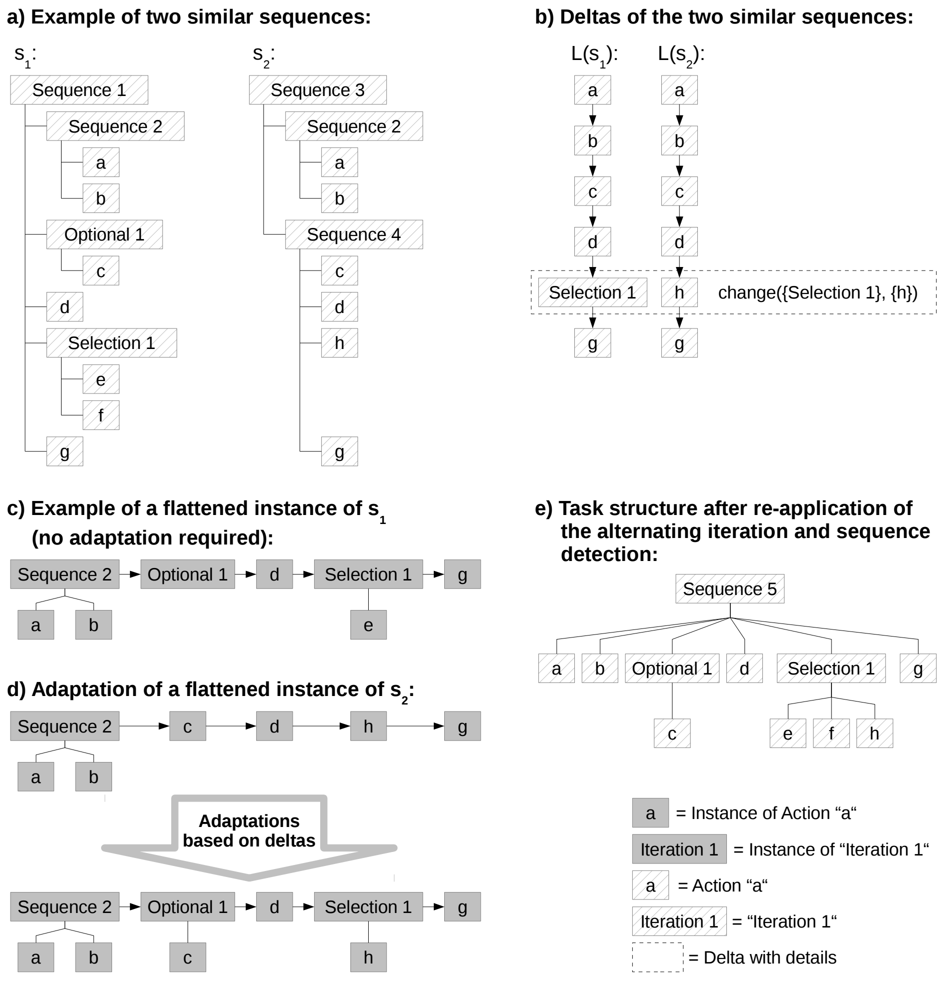

a) Example of two similar sequences:
s_1:
Sequence 1
Sequence 2
a
b
Optional 1
c
d
Selection 1
e
f
g
s_2:
Sequence 3
Sequence 2
a
b
Sequence 4
c
d
h
g
b) Deltas of the two similar sequences:
$L(s_1)$:
$L(s_2)$:
a
b
c
d
Selection 1
h
change({Selection 1}, {h})
g
c) Example of a flattened instance of s_1 (no adaptation required):
Sequence 2
Optional 1
d
Selection 1
g
a
b
e
d) Adaptation of a flattened instance of s_2:
Sequence 2
c
d
h
g
a
b
Adaptations based on deltas
Sequence 2
Optional 1
d
Selection 1
g
a
b
c
h
e) Task structure after re-application of the alternating iteration and sequence detection:
Sequence 5
a
b
Optional 1
d
Selection 1
g
c
e
f
h
a = Instance of Action "a"
Iteration 1 = Instance of "Iteration 1"
a = Action "a"
Iteration 1 = "Iteration 1"
= Delta with details

Figure 4.10.: Example for merging two similar sequences containing selections and optionals (adapted from [21]).

sequence detection. To prevent discarding task instances, we adapt the creation of flattened instances. For this, we start by determining direct or indirect child tasks of the sequences s_1 and s_2, which are shared between s_1 and s_2. Each of these tasks represents a sublist in $L(s_1)$ and $L(s_2)$. We then check for each shared child task if its corresponding sublists in $L(s_1)$ and $L(s_2)$ are at the same positions. If so, we do not flatten the instances of this task when creating of the flattened instances of s_1 and s_2. As a result, the flattened instances contain instances of already detected tasks, which do not get discarded on the subsequent iteration and sequence detection. Through this, we also ensure that during the merging process, no new iteration or sequence is detected, which is structurally identical to a previously detected iteration or sequence. An example for this preservation is shown in Figure 4.10 c and 4.10 d. There, the instances of *Sequence 2* are not flattened, because *Sequence 2* is a shared child between s_1 and s_2 (see Figure 4.10 a), and because it represents the same entries in $L(s_1)$ and $L(s_2)$ (see Figure 4.10 b). We use the same approach for child tasks of s_1 and s_2, whose corresponding sublists in $L(s_1)$ and $L(s_2)$ represent parts of a delta or a full delta. Through this, also the instances of these child tasks are preserved.

When merging two sequences s_1 and s_2, which have optionals as direct or indirect children, the creation of flattened instances also needs to be adapted. This is because an optional instance may not have a child, which indicates that the corresponding action was left out. In this situation, the optional instance has no flattened representation except itself. When creating the flattened instances of s_1 and s_2, we do not flatten instances of an optional being left out in instances of s_1 or s_2. An example for this is shown in Figure 4.10 c, which shows a flattened instance for the first Sequence s_1 in Figure 4.10 a. As the instance of *Optional 1* is left out, it is not flattened. The flattened instances of a similar sequence, that do not contain the optional, will contain the instances of the actions covered by the optional, instead. An example for a sequence similar to s_1 is s_2, also shown in Figure 4.10 a. A flattened instance of s_2 is shown in the upper part of Figure 4.10 d. It does not contain instances of *Optional 1*, as this does not belong to the task model. Instead, it contains instances of the actions, which are similar to the actions covered by *Optional 1*, i.e., an instance of action c. As in this case the flattened task instances are not harmonized anymore, we reharmonize them before we apply the subsequent iteration and sequence detection. We do this by creating optional instances and integrating them into the flattened instances, where the optional instances are not present. The optional instances replace the elements of the flattened instances, which need to be considered optional. For the example in Figure 4.10, this is shown for *Optional 1* in the lower part of Figure 4.10 d. There, the instance of action c is replaced by an instance of *Optional 1*, which has the replaced action instance as its child.

We also make exceptional considerations for selections. Selections are added directly to task lists as described in the previous section. This results in the fact, that two similar sequences s_1 and s_2 either share the same selection, or a selection is part of a delta, when considering $L(s_1)$ and $L(s_2)$. If s_1 and s_2 share the same selection z, we do not flatten respective instances, because z is a shared task and instances of shared tasks stay unflattened as described above. If a selection is part of a change delta or belongs to an insert or delete delta, the situation is handled as described in Section 4.4.3.3, i.e., no selection-specific handling is done. If a selection z fully covers one side of a change delta between two sequences s_1 and s_2, then the respective other side of the delta becomes a new variant of z. Let s_1 be the sequence containing z. When creating the flattened instances of s_1, the instances of z are not flattened. When creating the flattened instances of s_2, the action instances representing the other side of the change delta are replaced by an instance of z. This instance gets the replaced action instances as its children, introducing an intermediate sequence, if required. An example for this situation is shown in Figures 4.10 c and 4.10 d. There, *Selection 1* belonging to s_1 is one side of a change delta. Its instances being children of instances of s_1 are not flattened. In a flattened instance of s_2, the instance of *Action h*, which is the other side of the change delta, is replaced by a new instance of *Selection 1*, which has the instance of *Action h* as its child.

A change delta may also have one selection representing one side and another selection representing the other side. In this situation, both selections are merged. For this merge, we create a new selection, which gets the children of both selections as its children.

A final exceptional situation, that we consider, are interleaving iterations. Our sequence comparison so far compares only task lists. It does not consider the structure of two sequences, especially iterations inside. But iterations in sequences may cause, that different parts of the task lists of two sequences may be repeatable. For example, consider two sequences s_1 and s_2, where $L(s_1)$ consists of the actions a, b, and c, and $L(s_2)$ contains the actions a, d, and c. Let s_1 contain an iteration i_1, that allows $\{ab\}$ to be repeated multiple times, whereas s_2 contains an iteration i_2, which allows a repetition of $\{dc\}$. We call such iterations *interleaving iterations*. In this situation, our above approach would try to create a selection z between the actions b and d. But during the flattening of s_1 and s_2, it would create completely unharmonized instances if an instance of s_1 and an instance of s_2 contain multiple executions of i_1 and i_2. For example, a valid flattened instance of s_1 is $\{a'b'a'b'c'\}$, where i_1 is executed two times. This would be adapted to $\{a'z'a'z'c'\}$. In addition, a valid flattened instance of s_2 is $\{a'd'c'd'c'\}$, where i_2 is executed two times. This would be adapted to $\{a'z'c'z'c'\}$. Based on these unharmonized flattened instances, no correct iteration and sequence detection can be performed and, therefore, no single sequence as replacement for s_1 and s_2 can be detected. To prevent this, we detect interleaving iterations, that are executed multiple times in instances of s_1 and s_2, before performing the merge. We then create new task lists $L(s_1)$ and $L(s_2)$, which contain the interleaving iterations instead of the repeated actions. Based on them, we get a new set of deltas. When afterwards flattening the sequence instances, we do not flatten instances of the interleaving iterations. Furthermore, we adapt the flattened instances only on the new set of deltas. Through this, the flattened instances are harmonized and the interleaving iterations are preserved.

Due to interleaving iterations, $L(s_1)$ and $L(s_2)$ of two similar sequences s_1 and s_2 can become completely different, as similar elements are dropped. This is the case for the example in the previous paragraph. There, after the interleaving iterations i_1 and i_2 are detected and directly included in the task lists, $L(s_1)$ is $\{i_1\ c\}$ and $L(s_2)$ is $\{a\ i_2\}$. In these cases, the resulting sequence similarity will also become 0 as $L(s_1)$ and $L(s_2)$ do not contain common elements anymore. If we detect this, we do not perform a merging of s_1 and s_2. This is because although they initially seemed similar, in fact, they are not.

4.4.3.9. Adaptations of the Harmonization of Parent Tasks

As described in Section 4.4.3.5, we update iterations, which have merged sequences as their children. In later merging cycles, merged sequences can also be children of optionals. Hence, we perform the same adaptations for optionals as done for iterations.

In addition, it may be the case, that for two sequences s_1 and s_2, which are merged into s_3, there is only one iteration i_1 having s_1 as its child, but no iteration i_2 having s_2 as its child. Performing the harmonization of parent tasks as described before, this can lead to a situation, where s_3 once occurs without a parent iteration and once with. Before merging, our task trees always include iterations of a sequence at any position if a sequence is iterated at least once. Therefore, we ensure, that, if for s_2 there is no parent iteration, but for s_1 there is one, then s_1 is simply replaced by s_3, but s_2 is replaced by an iteration of s_3.

Due to the merging of children and the harmonization of parent tasks, two distinct parent sequences may become similar, or even the same. This is handled in subsequent merging cycles, as these sequences have a high similarity level. Therefore, they are selected to be merged, as well.

4.4.4. Complexity Analysis

When recording users of websites, large amounts of action instances may be recorded. The generation of task trees out of these action instances can, hence, become time-consuming. Therefore, we perform a complexity analysis as an estimation of the performance of our approach for larger input data. We do this by first considering the individual steps that we take in our generation process. Then, we define a complexity for them. finally, we derive a complexity for the whole approach. The overall task tree generation approach is shown in Algorithm 4.3. In the following paragraphs, we describe each of the steps in the algorithm and their complexities.

Initially, we perform an alternating iteration and sequence detection (Lines 1 to 4 in Algorithm 4.3). The iteration detection itself can be done by reading the input task instance list once. During this read, we can store directly, which actions are repeated, as well as the positions of the repeated actions. This storing has a complexity of $\mathcal{O}(1)$, as the information can be stored, e.g., in an array or a list. Afterwards, the instances of the repeated actions are replaced through respective iteration instances, which requires at most a second read of the input data. Hence, the iteration detection itself has a complexity of $\mathcal{O}(n)$, where n is the number of processed action instances.

The complexity of the sequence detection is similar, but due to the choosing of sequences to be merged, its calculation requires more insight. Through a single read of the input task instance list, all n-grams can be determined, including their size and locations in the input data. The only boundary here is the size of the memory, as an action instance list of length n contains $\sum_{i=2}^{n-1} i = \frac{n(n+1)}{2} - (n+1)$ permutations of n-grams l' with a length $1 < |l'| < n$. During the single read, the detected n-grams can directly be combined into n-gram sets, which represent the same action combinations. The assignment of n-grams to n-gram sets can be done with a complexity of $\mathcal{O}(1)$. For this, we can use an algorithm that is capable of using an n-gram as unique index of the corresponding n-gram set in an array of n-gram sets. From all n-gram sets, we determine which n-gram sets need to be replaced first. We initially choose those n-gram sets with first the highest number and second the longest length of n-

Algorithm 4.3 Simplified task detection process with complexities.

```
 1: repeat
 2:     // iteration detection                                      $\mathcal{O}(n)$
 3:     // sequence detection                                       $\mathcal{O}(n)$
 4: until no new sequence or iteration detected                     $\mathcal{O}(n^2)$
 5:
 6: repeat
 7:     // compare sequences                                        $\mathcal{O}(n^4)$
 8:     // choose sequences to be merged first                      $\mathcal{O}(n^2)$
 9:
10:     if there are sequences to be merged then
11:         for all sequence pairs to be merged do
12:             // adapt flattened task instances                   $\mathcal{O}(n^2)$
13:
14:             repeat
15:                 // iteration detection on flattened task instances  $\mathcal{O}(n)$
16:                 // sequence detection on flattened task instances   $\mathcal{O}(n)$
17:             until no new sequence or iteration detected         $\mathcal{O}(n^2)$
18:
19:             // harmonize parent tasks                           $\mathcal{O}(n)$
20:         end for                                                 $\mathcal{O}(n^3)$
21:     end if
22: until no sequences to be merged                                 $\mathcal{O}(n^5)$
```

grams. This can be done during the initial reading of the input data. When having read the next action instance from the input data, several new n-grams are completed. For each of these n-grams, we know to which of the n-gram sets it belongs. We also know the current number of n-grams in a set, as well as the n-gram length. Hence, we can also determine at any time during the first read, which n-gram sets currently contain most n-grams, and which length these n-grams have. This can be done by maintaining pointers to these n-gram sets. These pointers are also available at the end of reading the input data. Hence, the initial choosing of the n-gram sets, which contain most and the longest n-grams, also has a complexity of $\mathcal{O}(1)$. After this initial choosing, only an amount of n-gram sets remains, which is a fraction of n. This is because a task instance list of length n contains only $n+1-|l'|$ permutations of n-grams l' of length $|l'|$. These n-grams must be identical to be added to an n-gram set. As the minimum number of identical n-grams per set is two, at most $\frac{n+1-|l'|}{2}$ n-gram sets will remain after the first choosing. The subsequent choosing process will discard at least one of the sets in any repetition and, therefore, runs at most $\frac{n+1-|l'|}{2}$ times. This shows, that considering an unlimited amount of memory, also the sequence detection has a maximum complexity of $\mathcal{O}(n)$.

The iteration and sequence detection are repeated alternately (Lines 1 to 4 in Algorithm 4.3). Either, the iteration or the sequence detection will detect at least one iteration or one sequence in a cycle. In the worst case scenario, only two action or task instances in the task instance list are combined to one new task instance per cycle, resulting in at most $n-1$ cycles for the iteration and sequence detection. Hence, the complexity of the alternation is also $\mathcal{O}(n)$. As the iteration and sequence detection already have a complexity of $\mathcal{O}(n)$, and as they are called in the alternation, the resulting complexity of the iteration and sequence detection is $\mathcal{O}(n^2)$.

After the iteration and sequence detection, we perform a merging of similar sequences (Lines 6 to 20 in Algorithm 4.3). For this, we first have to determine similar sequences, which is done by comparing each sequence with any other sequence. This means performing $n(n-1)$ comparisons if n is the number of sequences. Considering an algorithm for comparing two sequences with a complexity of at most $\mathcal{O}(n^2)$, which is given for Myers Diff algorithm used in this thesis [89], the complexity of the detection of similar sequences is $\mathcal{O}(n^4)$. Afterwards, we perform a choosing of similar sequences to be merged first. The complexity of this choosing increases with an increasing number of sequences that are shared between the pairs. But still any pair is handled at most three times, resulting also in a maximum complexity of $\mathcal{O}(n^2)$ where n is the number of sequences.

The subsequent flattening of task instances needs to be done for any task instance and also for any detected delta. Hence, the complexity of this step is linearly dependent on the multiplicity of the number of task instances and the number of deltas. Therefore, it is at most $\mathcal{O}(n^2)$. Afterwards, we perform an alternating iteration and sequence detection on the flattened instances, which, as shown above, also has a complexity of $\mathcal{O}(n^2)$. Furthermore, we do a harmonization of parent tasks, which can be done in linear time ($\mathcal{O}(n)$). Finally, after a merging, we check if there are further similar sequences and merge them if required. Through this repetition of the merging process, the complexity increases to $\mathcal{O}(n^5)$ for the overall merging process. So referring to Algorithm 4.3, the major complexity issue is the multiple repetition of Line 7, which already has a complexity of $\mathcal{O}(n^4)$.

The complexity of our approach with $\mathcal{O}(n^5)$ seems to be quite high. But this considers the worst case scenario, in which we have as few as possible identical n-grams in the task instance list. The complexity also depends strongly on the number of distinct available actions and action combinations. If, for example, the user can perform only a small set of distinct actions and only a small amount of distinct action combinations, even large task instance lists will contain many identical n-grams. Hence, already the sequence and iteration detection will perform many more reductions of the number of entries in the task instance list within one cycle, than done in the worst case scenario. The same applies for the detection and merging of similar sequences. With a decreasing number of possible action combinations, fewer similar sequences are detected and merged. In addition, after a certain data set size is reached, no new actions or action combinations are recorded, because the previously recorded data already contains all actions and action combinations possible on a software. Hence, the number of iteration and sequence detection cycles, as well as the number of required sequence comparisons, do not increase anymore, even if the data set size increases. Furthermore, action combinations performed only once in a data set are not detected and, hence, there is no corresponding detection cycle for them in the iteration and sequences detection.

Additionally, the approach has some opportunities for runtime improvements, which are not considered in the complexity analysis. For example, one aim in this thesis is to determine, how many action instances a detected sequence should cover to consider it representative for typical user behavior. This implies not detecting sequences, that are not be representative. Through this, we intent to reduce the number of cycles in the iteration and sequence detection, as well as the amount of sequence comparisons for the subsequent merging. In fact, in our case studies, we only compare and merge those sequences with each other that cover most of the recorded action instances and which are, hence, the most representative ones. Furthermore, when comparing sequences for detecting similar ones, it is possible to skip comparisons based on the knowledge about the sequences structures. For example, if the task list of one sequence is much longer than the one of another sequence, the similarity level $sim(s_i, s_j)$ can be estimated to be lower than sim_{min}. In this case, the comparison does not need to be done. We also skip sequence pairs whose deltas are at the beginning or end of the task lists. This can also be checked before applying a complex diff algorithm on the task lists. Hence, it is sufficient to only compare sequences with each other whose first and last elements of the corresponding task lists are the same. In addition, if a sequence s_1 is the direct or indirect child of a sequence s_2, the comparison of s_1 and s_2 can be skipped, because parent tasks are not merged with their children. This is ensured by the choosing process applied for identifying sequences to merge. All this reduces the number of required comparisons significantly. Finally, the remaining comparisons can be done in parallel, as they are independent from each other, which can further reduce the runtime. We implemented the optimizations named in this chapter in our case studies.

4.5. Usage and Task-Tree-Based Usability Evaluation

In this section, we describe our approach for usability smell detection. We start by introducing the basic concepts. Then, we explain the detection of usability smells of two separate major types. The first type are usability smells with relation to the generated task trees. The second type are usability smells, which are detected solely based on recorded action instances.

4.5.1. Approach

The detection of usability smells as done in our approach results in two smell categories. The usability smells of the first type are detected using the generated task trees. Most of these smells have a direct relation to a specific task and provide task specific details. The usability smells of the second type are detected solely based on the recorded action instances. They have a more statistical nature and refer mostly to specific elements of a GUI, such as a specific text field or a specific view. For each of the smells, we provide

- a foundation and a justification for the smell,
- expected user behavior if the corresponding usability issue is present,
- expected occurrence of the user behavior in the task trees or the recorded action instances,
- a description of the detection of the smell, including the calculation of its intensity,
- the possible usability critique the smell can provide, and
- if applicable, a simple example for the smell.

In the foundation and justification for each smell, we provide references to the corresponding literature. Most important here is the guideline catalog "The Research-Based Web Design & Usability Guidelines, Enlarged/Expanded edition" of the U.S. Dept. of Health and Human Services [90]. Although being from 2006, this catalog includes many guidelines for user interface design, which are still valid today. In the catalog, each guideline is described and rated for its validity based on available studies and literature as well as based on experiences of known usability experts. Hence, the guidelines have a scientific background, which is described in the introduction section of the catalog.

The intensity of each smell is given as a smell specific calculation. For each smell, we provide a formula used for the calculation of the intensity. The intensity of a specific *smell* found for a task t is given as $R_{smell}(t)$. The intensity of a specific *smell* found for a task instance t' is given as $R_{smell}(t')$. The intensity of other smells is simply named as R_{smell}. We do not provide smell specific intensity thresholds, which need to be exceeded to consider a smell present. This is due to the fact, that we aim at determining possible thresholds in the case studies of this thesis.

For the smells, we provide examples for an improved clarification of their detection and intensity calculation. For some smells, it is not possible to provide simple examples. The reason is, that even a simple example for a specific smell may include a large number of tasks, actions, or respective instances. Hence, we provide the examples only for those smells, where the examples are rather simple.

4.5.2. Detection of Usability Smells Based on Task Trees

In this section, we describe the usability smells that have a relationship to the generated task trees. Either, the smells refer to a single task or they consider all detected tasks in general. Most smells focus on tasks of type sequence, only. This is because sequences express the main execution order of a task, whereas iterations, optionals, and selections only express execution variants. We already published some usability smell descriptions in this section in a similar fashion in [22]. We indicate for the respective smells if this is the case. The smells described in this section are

- Important Tasks,
- Required Inefficient Actions,
- High GUI Element Distance,
- Missing Feedback,
- Required Input Method Change, and
- Missing User Guidance.

4.5.2.1. Important Tasks

We already published a similar description of this smell in [22].

Foundation and Justification One important aspect of usability is efficiency [28]. Especially tasks executed often should be executed with a high efficiency [90]. For example, they should require only a minimum of actions [82, 91, 1]. This can be achieved by automating specific action combinations for a task, if applicable [15].

Expected User Behavior The most important tasks performed with a software are executed most often by users. In addition, the more often users perform the same task, the more similar are the performed actions and their respective order.

Expected Occurrence in Generated Task Trees As the task tree generation generates tasks for those action combinations executed most often by the users, regularly executed action combinations are represented as tasks with many instances in the task trees.

Detection and Intensity The detection focuses on sequences only. For each sequence s, we determine the action instances $a'(s)$ from which s was detected. We correlate this to the number of all recorded actions instances $|A'|$. As a result, we get the following ratio as the smell's intensity:

$$R_{impTask}(s) = \frac{a'(s)}{|A'|} \tag{4.5.1}$$

This ratio is the percentage of recorded action instances, which represent instances of s in all recorded action instances. The higher the intensity $R_{impTask}(s)$, the more important is s for the users when using a software.

Possible Usability Critique The smell provides an ordering of the detected user tasks by their importance, when regarding the number of recorded action instances. It can directly state, which tasks are of most importance for the user, and propose to optimize the executions of these tasks to increase the user's efficiency. Through the relationship to respective tasks, the smell provides much information helpful for optimizing the task. This includes the actions belonging to the task as well as the typical task executions. However, the smell can not give more detailed information about how the optimization should be done.

4.5.2.2. Required Inefficient Actions

We already published a similar description of this smell in [22].

Foundation and Justification When executing tasks, users perform inefficient actions, which have no or minimal semantical effect on the task fulfillment [80]. An example is performing the scroll action when filling out a form. In some situations, these actions may be required for advancing the task execution. For example, if a form is too large, users need to scroll to reach the next element to be filled out. As required inefficient actions decrease the users efficiency, they should be minimized [90].

Expected User Behavior If inefficient actions are not required, users will perform them in an unsystematic way. But if inefficient actions are required for the execution of a specific task, users will perform them as needed and systematically.

Expected Occurrence in Generated Task Trees The task trees are generated based on all recorded action instances, including inefficient ones. Therefore, they include inefficient actions at the positions where they were executed and especially where they are required. In addition, the task instances include the instances of inefficient actions if they were performed, even if in the task model they are optional.

Detection and Intensity The detection of this smell focuses on sequences only. For each instance $s' \in x(s)$ of a sequence s, we first determine the number of inefficient actions $ia'(s')$ performed in the instance. Then, we calculate the intensity of the smell as follows:

$$R_{ineffAct}(s) = \frac{1}{|x(s)|} \sum_{s' \in x(s)} \frac{ia'(s')}{|a'(s')|} \tag{4.5.2}$$

This ratio is the percentage of inefficient action instances performed on average, when executing sequence s. The higher the intensity $R_{ineffAct}(s)$, the more inefficient actions are performed when executing s on a software.

Possible Usability Critique The smell detects tasks that are usually executed with a certain ratio of inefficient actions. Due to the task relationship, the smell can provide detailed information about where and when the inefficient actions seem to be required and which inefficient actions are performed. Therefore, it can also propose, which actions should be prevented for the task execution. How the actions can be prevented, depends on the type of action. Actions of type scrolling are typically an indicator, that the visual screen space is too small for displaying a certain view. Hence, the view must become smaller to be displayable at once. This can be proposed by the smell.

Example An example for a task structure where the smell has a high intensity is shown in Figure 4.11. The figure displays two instances of an example *Sequence 1*. Both instances have two action instances as their children, of which one is an inefficient scrolling action and the other is an efficient click on a button. For *Sequence 1*, the smells intensity $R_{ineffAct}(Sequence\ 1)$ is 0.5, i.e., 50%, indicating that half of the actions done for executing *Sequence 1* are inefficient actions.

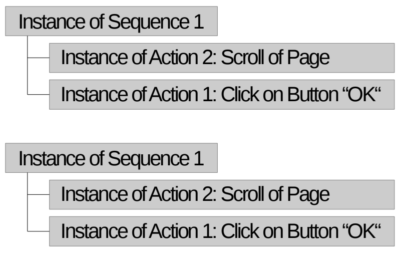

Instance of Sequence 1
Instance of Action 2: Scroll of Page
Instance of Action 1: Click on Button "OK"
Instance of Sequence 1
Instance of Action 2: Scroll of Page
Instance of Action 1: Click on Button "OK"

Figure 4.11.: Example for the usability smell "Required Inefficient Actions" (adapted from [22]).

4.5.2.3. High GUI Element Distance

We already published a similar description of this smell in [22].

Foundation and Justification The structure of a GUI should allow for an efficient execution of a task [90]. This can be achieved through an appropriate arrangement of GUI elements. This means, that GUI elements required for the execution of a task should be arranged in the order, in which they are used, and they should be as colocated as possible [82].

Expected User Behavior Users interact with GUI elements in the order required for the task executions.

Expected Occurrence in Generated Task Trees The task trees show the order, in which actions are performed to fulfill a task. As actions refer to GUI elements, the task trees also show the used GUI elements and their usage order.

Detection and Intensity The detection of this smell is based on a metric for the distance of GUI elements. In our work, the GUI models do not refer to physical locations of GUI elements. Hence, our distance metric is based on the tree structure of the GUI model. This structure contains information about container elements and views, to which individual GUI elements belong. We utilize this information for the distance metric. We consider different distances between two GUI elements. These distances depend on the elements' locations in the GUI element tree. We define the distance metric $dist(g_1, g_2)$ between two GUI elements g_1 and g_2 as follows:

$$dist(g_1, g_2) = \begin{cases} 0.0 & g_1 \text{ and } g_2 \text{ are identical} \\ 0.2 & g_1 \text{ and } g_2 \text{ have the same direct parent container element} \\ 0.5 & g_1 \text{ and } g_2 \text{ belong to the same view} \\ 0.75 & g_1 \text{ and } g_2 \text{ belong to the same software} \\ 1.0 & g_1 \text{ and } g_2 \text{ belong to different software} \end{cases} \tag{4.5.3}$$

The distance values of this metric are arbitrary and could be chosen differently. The way we define them allows us to have smaller distances for closer GUI elements and larger distances for distant GUI elements. The unequal distribution between the distances intends to put more weight on GUI elements, which are very close to each other, i.e., in the same panel. The highest distance is 1.0, which indicates that two GUI elements do not belong to the same software.

Based on this metric, we perform the detection of the usability smell. We focus on sequences only. We first determine for any instance s' of a sequence s the action instances that were performed. From them, we extract the list of GUI elements utilized by the action instances. From this list, we remove the GUI elements referred to by instances of inefficient actions, as these actions do not contribute to the semantic fulfillment of the task. The result is a list $G(s') = g_1 \dots g_n$ of GUI elements utilized by instances of efficient actions in the sequence instance s'. The sorting order of the list is the order, in which the GUI elements were used. We then calculate for any subsequent pair of GUI elements in $G(s')$ the distance and sum up the distances to the cumulative distance $\sum_{i=2}^{|G(s')|} dist(g_{i-1}, g_i)$ of GUI elements utilized in s'. Furthermore, we determine the number of distances, which is $|G(s')| - 1$. Finally, we determine the average distance between two subsequently used GUI elements in instances of sequence s. This average distance is the intensity of the smell. For its calculation, we sum up all cumulative distances and divide them by the sum of the number of distances. The corresponding formula for the smell intensity is as follows:

$$R_{highDist}(s) = \frac{\sum_{s' \in x(s)} \sum_{i=2}^{|G(s')|} dist(g_{i-1}, g_i)}{\sum_{s' \in x(s)} (|G(s')| - 1)} \tag{4.5.4}$$

This ratio is the average distance between two subsequently used GUI elements, when executing efficient actions for performing task s. The higher the intensity $R_{highDist}(s)$, the higher the distance between these GUI elements, what decreases the efficiency of the task execution. The intensity values can be traced back to the distance metric $dist(g_1, g_2)$. For example, if $R_{highDist}(s)$ is higher than 0.5, then, when executing the efficient actions of s, two subsequently used GUI elements are located in different views of the software.

Possible Usability Critique Through the relationship to a concrete task, this smell provides detailed information about which GUI elements are used, as well as the usage order. If the distances between the required GUI elements are large, the smell gives concrete information about which GUI elements should be more colocated to improve the users efficiency for executing the task. The smell also shows the order, in which the GUI elements should be arranged for best supporting the task execution.

Example An example for a task with a high intensity of this smell is shown in Figure 4.12. On the left side, this figure contains a GUI model of a website with two pages. On *page 1*, there is a link with id *link1* leading to *page 2*, as well as a button with id *OK*. On *page 2*, there is a link with id *link2* leading to *page 1*. On the right side, the Figure shows an instance of a task *Sequence 2*. It represents the executions of a click on *link1*, a click on *link2*, and finally a click on button *OK*. Therefore, $G(\mathit{Instance\ of\ Sequence\ 2})$ is $\{link1, link2, OK\}$. The distance between *link1* and *link2*, as well as between *link2* and the button *OK*, is 0.75

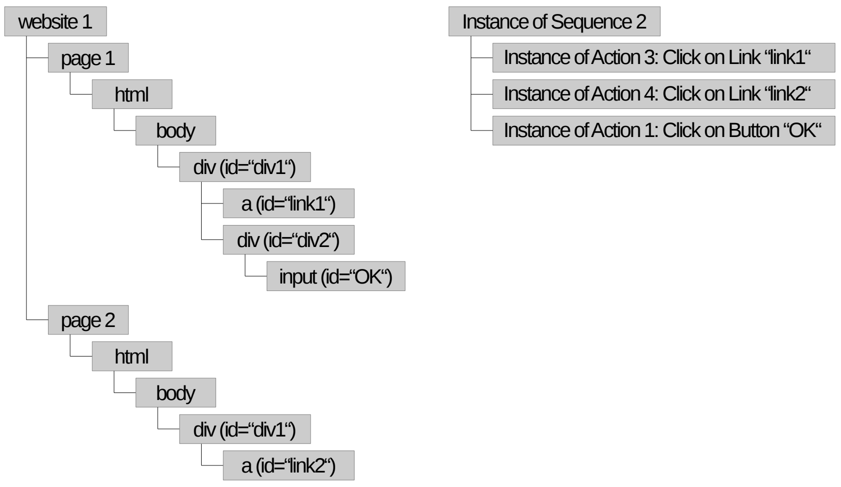


Figure 4.12.: Example for the usability smell "High GUI Element Distance" (adapted from [22]).

in our distance metric. The sum of these distances is 1.5. As this is the only instance of *Sequence 2*, the resulting intensity of the smell $R_{highDist}(Sequence\ 2)$ is $\frac{1.5}{2} = 0.75$.

4.5.2.4. Missing Feedback

We already published a similar description of this smell in [22].

Foundation and Justification For actions performed by users, a software must provide feedback to indicate that the action instance is being or was processed [92, 93, 94, 8, 90].

Expected User Behavior If a software does not provide any feedback as reaction on an action instance, or if the feedback is not recognized, users repeat the action one or more times, as it seems to them, that the previous action instance is not processed. For example, if users click on a download link of a website and nothing visible happens, they click the link again.

Expected Occurrence in Generated Task Trees Repetitions of an action indicating missing feedback are transformed into iterations of the action in the task trees. Not all repeated actions indicate missing feedback. For example, while reading a longer text on the screen, users may repeatedly scroll. Hence, in this smell we only consider repetitions of clicks on buttons and links. In addition, users may accidentally repeat a click. For example, they may perform a double click on a button, although a single click would be sufficient. Furthermore, regarding download links on websites, users may click a download link several times, each time for starting the download again. Usually, these repetitions occur after some time has elapsed. Therefore, we only search for repetitions that occur in a specific time frame. We consider repetitions in a time frame that is smaller than $1,000$ milliseconds as accidental. Furthermore, we consider repetitions in a time frame that is larger than $15,000$ milliseconds as intended.

Detection and Intensity For the detection of the smell, we filter the generated tasks for iterations of clicks on buttons and links. For any iteration i matching this criterion, we consider each instance $i' \in x(i)$. We determine the list of action instances $a'(i') = a'_1 \dots a'_{|a'(i')|}$ which represent i'. Of this list, we create sublists of a minimum length of two. In these sublists, the time stamp difference $\Delta ts(a'_i, a'_j)$ between the first action instance in the sublist a'_i and the last action instance in the sublist a'_j is $1,000 \leq \Delta ts(a'_i, a'_j) \leq 15,000$. The sublists may overlap in exactly one element. This means, that the last element of one sublist can be the first element of the next sublist. The sublists are as long as possible, to determine a maximum number of actions performed in a time frame smaller than $15,000$ milliseconds. The determined sublists do not necessarily cover all action instances in $a'(i')$. We do this sublist determination to ensure to have representations of repeated action executions that indicate a missing feedback.

For each of the determined sublists $a'_i \dots a'_j$, we calculate a measure for the missing feedback. The measure depends on the time $\Delta ts(a'_i, a'_j)$ between the first and the last action instance in a sublist, as well as on the length of the sublist being $j - i$. The missing feedback for a sublist is calculated as $\Delta ts(a'_i, a'_j) \times (j - i)$. Through this, the missing feedback measure increases with the time, but also, the more often a user clicks. Thus, the less often the user sees any feedback, the more increases the missing feedback measure. For the iteration instance i', the resulting missing feedback measure is then the sum of the missing feedback measures of the determined sublists, and it is calculated as follows:

$$R_{missFeed}(i') = \sum_{a'_i \dots a'_j \in a'(i')} \Big(\Delta ts(a'_i, a'_j) \times (j - i) \Big) \tag{4.5.5}$$

In this formula, a'_i and a'_j represent the first and the last elements of a determined sublist of $a'(i')$. This sublist matches the above mentioned criteria, which also implies $i < j$. The missing feedback measure for the iteration i, which is also the intensity of the smell, is then the average missing feedback measure for all instances $i' \in x(i)$. It is calculated as follows:

$$R_{missFeed}(i) = \frac{1}{|x(i)|} \sum_{i' \in x(i)} R_{missFeed}(i') \tag{4.5.6}$$

Although the unit of $R_{missFeed}(i)$ is milliseconds, it does not represent milliseconds. Instead, it increases with the number of repeated action executions if they are done in the predefined time range. The more often the action is repeated, the more increases $R_{missFeed}(i)$. If there are no repetitions of the respective action that fall in the time frame of 1 to 15 seconds, $R_{missFeed}(i)$ evaluates to 0ms. A value of 1000ms for $R_{missFeed}(i)$ for an iteration i that has 10 instances indicates, that in half of the instances the corresponding action is typically repeated once in a time frame of 2 seconds. $R_{missFeed}(i)$ evaluates to the same result if in only two of the instances of i, the corresponding action has been repeated twice in a time frame of 5 seconds.

Possible Usability Critique Through the relationship to the action, the smell provides detailed information about where the user might be missing feedback. The corresponding iteration and its instances give detailed information about how the action was repeated and, hence, gives the evaluator some room for interpretation. However, the smell cannot provide guidance on how to solve the corresponding issue in details. Nevertheless, it can provide respective ideas and possible patterns to be applied.

Example An example for a task with a high intensity of this smell is shown in Figure 4.13. It shows three instances of an iteration of *Action 1*, which is a click on a button. In the first instance, the action is performed twice, in the other instances only once. In the first instance, the figure also shows the time stamps of the action instances. The time difference between them is $4000\text{ms} - 1000\text{ms} = 3000\text{ms}$ milliseconds. Therefore, $R_{missFeed}(\mathit{Iteration\ 1})$ evaluates to $\frac{1}{3}(3000\text{ms} \times 1 + 0 + 0) = 1000\text{ms}$.

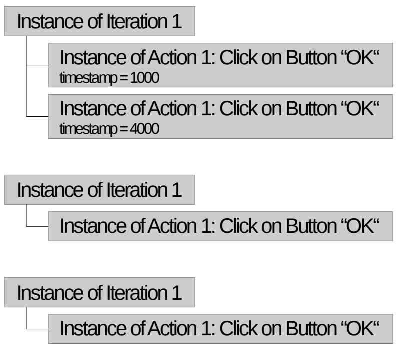

Instance of Iteration 1
Instance of Action 1: Click on Button "OK"
timestamp = 1000
Instance of Action 1: Click on Button "OK"
timestamp = 4000
Instance of Iteration 1
Instance of Action 1: Click on Button "OK"
Instance of Iteration 1
Instance of Action 1: Click on Button "OK"

Figure 4.13.: Example for the usability smell "Missing Feedback" (adapted from [22]).

4.5.2.5. Required Input Method Change

Foundation and Justification When executing tasks, users should not be forced to continuously switch between input methods [90]. For example, when filling out a form, users should not need to continuously switch between mouse and keyboard actions. Instead, the form should support both, filling out the form only with the keyboard and only with the mouse, so that the user has the choice. For text inputs, the second variant often cannot be achieved. But for example, entering a date into a text field can be supported by both, text input and mouse input, through a date chooser.

Expected User Behavior If required, users switch between input methods when executing a task.

Expected Occurrence in Generated Task Trees The detected tasks and their instances show the order in which actions are executed. The actions themselves define the input method used for performing them. Hence, based on the tasks and their instances, it can be determined which input method changes users need to do to perform a task.

Detection and Intensity The detection of this smell focuses on sequences only. We first determine for any instance s' of a sequence s the list $a'(s')$ of action instances that were performed. We then determine for any subsequent pair of action instances if they have a distinct input method, e.g., if one is an instance of a mouse action and the other an instance of a keyboard action. Then, we sum up the number of input method changes to $inputMethodChanges(s')$ and divide it by the number of all possible input method changes in s', which is $|a'(s')| - 1$. Finally, we determine the average ratio of input method changes for instances of sequence s, which is the intensity of the smell. For this, we sum up all input method change ratios of the instances $s' \in x(s)$ and divide them by the number of instances of s. The corresponding formula for the smell intensity is as follows:

$$R_{inputMethodChange}(s) = \frac{1}{|x(s)|} \sum_{s' \in x(s)} \frac{inputMethodChanges(s')}{|a'(s')| - 1} \tag{4.5.7}$$

$R_{inputMethodChange}(s)$ is the average ratio of input method changes required for two subsequent action instances in the instances of s. This ratio increases, the more input method changes are performed. At most, it can be 1, i.e., 100%, if for any two subsequent action instances an input method change is done. It is 0, i.e., 0%, if no input method change is done.

Possible Usability Critique Through the reference to the task, the smell can provide detailed information about action combinations that require input method changes. Through the actions, it can also provide detailed information about the involved GUI elements. The smell cannot provide detailed guidance on how to solve the corresponding usability issue. But it can provide patterns, which can be applied to minimize input method changes.

Example An example for a task with a high intensity of this smell is shown in Figure 4.14. It shows an instance of a sequence covering five action instances. The first, third, and fifth action instances are mouse clicks, the second and fourth action instances are text inputs. Between all action instances, a change between mouse and keyboard is required. This sums up to four input method changes. Therefore, $R_{inputMethodChange}(Sequence\ 3)$ evaluates to $\frac{1}{1}(\frac{4}{5-1}) = 1$, being the highest possible intensity for this smell.

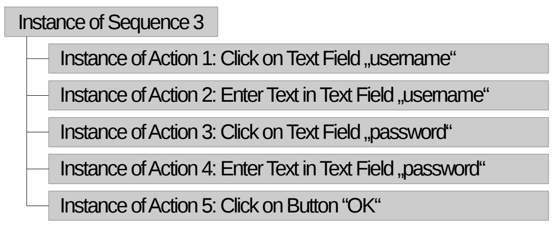

Instance of Sequence 3
Instance of Action 1: Click on Text Field „username“
Instance of Action 2: Enter Text in Text Field „username“
Instance of Action 3: Click on Text Field „password“
Instance of Action 4: Enter Text in Text Field „password“
Instance of Action 5: Click on Button “OK“

Figure 4.14.: Example for the usability smell "Required Input Method Change".

4.5.2.6. Missing User Guidance

Foundation and Justification Users require guidance when using a software [94]. If they do not have guidance when, e.g., searching an information or filling out a form, they tend to pogo sticking [95, 96], i.e., trying things and going back, or corrections when entering data [82].

Expected User Behavior The less guidance users have, the more they try things out. This means, they perform actions in a large number of different combinations instead of only some common ones.

Expected Occurrence in Generated Task Trees The more common action combinations users perform, and the more guidance they have, the less distinct tasks are generated. In addition, most recorded action instances belong to instances of detected tasks.

Detection and Intensity For the detection of this smell, we consider all task instance lists that are the result of the task detection. These lists contain instances of actions and detected tasks. An instance of a detected task represents several action instances. Hence, m elements in a task instance list represent n action instances, where $m <= n$. The more action instances are covered by instances of detected tasks, the smaller is m and the larger is the difference between m and n. The smaller m in comparison to n, the more common tasks were performed by the users and the more guidance they had.

The goal of this smell is to determine an average number of m for any combinations of $n = 10$ subsequently performed actions. This gives an impression of the average ratio between m and n. For this, we define the multiset A^{10} which contains all combinations of 10 subsequently performed actions a^{10}. It also contains as many duplicates of an a_i^{10} as often as it was performed by the users. We then map each $a_i^{10} \in A^{10}$ to its representation in the task instance lists, as they resulted from the task detection. Duplicates of an a_i^{10} are mapped to different representations to ensure, that all different occurrences of the respective action combination are covered. Then, for any $a_i^{10} \in A^{10}$, we determine $m(a_i^{10})$, which is the number of elements in the task instance lists that represent a_i^{10}. We then determine the average of $m(a_i^{10})$ for any $a_i^{10} \in A^{10}$. The result is the intensity of the smell, whose formula is as follows:

$$R_{guidance} = \frac{1}{|A^{10}|} \sum_{a_i^{10} \in A^{10}} m(a_i^{10}) \tag{4.5.8}$$

The value of $R_{guidance}$ is higher, the more elements of a task instance list on average represent the execution of 10 subsequent actions. Its minimum value is 1 indicating that on average, one element in a task instance list represents 10 subsequently executed actions. Its maximum value is 10, meaning on average each executed action has its own representation in the task instance lists resulting from the task detection. The more common tasks are executed by users, the more action instances are covered by a single entry in the task instance lists and, hence, the smaller $R_{guidance}$.

Possible Usability Critique The smell has no reference to an individual task and does, therefore, not provide details on task level. However, it provides a summary information, similar to the average number of errors performed by users. This helps an evaluator to get an overall impression of the user guidance of a software. Hence, the software can be critically analyzed and improved in this respect. The smell can provide heuristics for possible improvement, e.g., a hint to adapt the terminology of the software to the terminology known by the users. It cannot provide details about where a heuristic is violated.

4.5.3. Detection of Usability Smells Based on Action Instances

In this section, we describe the usability smells which are solely detected based on recorded action instances. These are

- Required Text Format,
- Text Input Repetitions,
- Text Input Ratio,
- Single Checking of Checkboxes,
- Misleading Click Cue,
- Required Text Field Focus,
- Good Defaults, and
- Unused GUI Elements.

4.5.3.1. Required Text Format

Foundation and Justification When entering text, users should not be forced to enter specific text formats and they should be supported to ease the entering process [91, 97, 98, 90].

Expected User Behavior If required, users enter text into a text field in a specific format. If the format is too complex or not understood, users will make errors and retype into the text field, until the required format is matched.

Expected Occurrence in Recorded Action Instances The recorded action instances will contain text entered into text fields. If a specific format is required, the text will usually include special characters, e.g., dashes, slashes, colons, and dots, that make up the required format.

Detection and Intensity The detection of this smell focuses on instances of text input actions. It searches the recorded data for respective action instances and extracts the texts that are entered into specific text fields. Then, for each text field, it determines the number $n_{noLetterOrDigit}$ of non-letter and non-digit characters that were entered into the text field in all text input action instances. It relates this number to the number n_{all} of all characters that were entered into the text field. The result is the ratio between both numbers, which is the following intensity of the smell:

$$R_{textFormat} = \frac{n_{noLetterOrDigit}}{n_{all}} \tag{4.5.9}$$

The value of $R_{textFormat}$ is higher, the more special characters users enter into a specific text field. It can become at most 1, i.e., 100%, indicating the only special characters are entered into the text field. It is 0, i.e., 0% if only letters or digits are entered into the text field.

Possible Usability Critique The smell refers to a concrete text field. Hence, it can give detailed information about into which text field many special characters are entered, usually.

Example An example for the detection of this smell are text fields, into which dates must be entered. Dates typically follow a specific format. A date in the German notation, e.g., is in the format *24.12.2015*, where the first two digits are the day, followed by a dot, and two digits for the month, followed by a dot, and four digits for the year. The special characters here are the two dots. If a user enters a correct date into this text field, then on any entering of the ten characters, two of them are no letter or digit. Hence, for such a text field, the intensity of the smell is $R_{textFormat} = \frac{2}{10} = 0.2$. Thus, 20% of the entered characters are no letters or digits.

4.5.3.2. Text Input Repetitions

Foundation and Justification When entering text, users should not be forced to enter the same data several times or to remember data displayed in one view that needs to be entered in another view [94, 90].

Expected User Behavior If required, users will reenter the same text in separate text fields if the same text needs to be filled in.

Expected Occurrence in Recorded Action Instances The recorded action instances include text entered into text fields. If the same text is entered into distinct text fields in the same user session, then the session includes several text input action instances, which were performed on different text fields, but in which the same text was entered.

Detection and Intensity The detection of this smell analyses the entering of text into text fields. For each text field, we first determine the number of all texts $n_{allTexts}$ entered in the text field. Then, we determine for any other text field if there is a text, which was entered into both text fields in the same user session. This results in the number $n_{reenteredTexts}$ of all texts that were entered into both text fields in the same user session. We then set both numbers into relation, resulting in the following ratio, which is the also the smell's intensity:

$$R_{textRepetitions} = \frac{n_{reenteredTexts}}{n_{allTexts}} \tag{4.5.10}$$

$R_{textRepetitions}$ always refers to a pair of text fields. A value of 1 indicates that in 100% of the usages of the first text field, the same text is entered into the second text field in the same user session, as well. A value of 0 indicates, that never a text, which is entered into the first text field, is also entered into the second text field in the same user session.

Possible Usability Critique As the smell refers to text field pairs, it can provide detailed information about into which text fields usually the same text is entered. In addition, it can inform about the repeated texts. The smell can only be detected if the entered text is recorded or encrypted in a way, so that subsequent text comparisons are possible.

4.5.3.3. Text Input Ratio

Foundation and Justification When entering data, users should not be forced to enter too much text into text fields [91, 90]. Instead, data should be entered via data type specific GUI elements, e.g., date choosers for entering a date.

Expected User Behavior If required, and if not possible otherwise, users will enter data by entering text into text fields.

Expected Occurrence in Recorded Action Instances The recorded action instances include text, which is entered into text fields and text areas, as well as the usage of other data entry methods.

Detection and Intensity For the detection of this smell, we first count the number of action instances $n_{textInput}$, which are text inputs into text field and text areas. Then, we set this value in relation to all recorded action instances, which results in the following intensity of the smell:

$$R_{textInputRatio} = \frac{n_{textInput}}{|A'|} \tag{4.5.11}$$

$R_{textInputRatio}$ varies between 0 and 1. 0 indicates no text inputs and 1 indicates that 100% of the action instances were text inputs. Especially, for devices on which text inputs are hard to perform, $R_{textInputRatio}$ should strive for 0.

Possible Usability Critique The smell does not refer to a specific action or GUI element. Hence, it only provides a general statistic about the usage of the system, as well as the possibilities of the system for entering data.

4.5.3.4. Single Checking of Checkboxes

Foundation and Justification User interfaces should provide GUI elements for entering data, that match the type of entered data. For example, a GUI should offer radio buttons instead of check boxes if no multiple checking of options is useful or required [90].

Expected User Behavior If of a set of check boxes, always only one option is relevant for users, then users check only one option any time the set of check boxes is displayed.

Expected Occurrence in Recorded Action Instances The recorded action instances have a reference to the GUI elements on which they were observed. Through their child parent relationship, GUI elements themselves refer to the view to which they belong. Hence, we can determine $viewActionInstances(A', view)$, which is the lists of subsequent recorded action instances that took place in the same view. If users always check only one of several check boxes of a check box group during the displaying of a view, then all lists in $viewActionInstances(A', view)$ contain a final checking of only one check box of a check box group.

Detection and Intensity To detect this smell, we first determine groups of check boxes that semantically belong together. Normally, a GUI model does not include directly, which check boxes belong together. To cope with this, we apply a heuristic. In this heuristic, we determine all check boxes referenced by action instances. Then, we group these check boxes. For this, we determine a list of the paths to the check boxes through the GUI model. These paths contain the parent elements of the check boxes and end with the check boxes themselves. Then, we order the path list, so that those paths being most similar, i.e., that contain as many as possible identical parent elements, lie next to each other in the list. Afterwards, we search the list for adjacent paths that differ at most in three elements. Due to the ordering of the list, the differences are in the last three elements, of which one are the check boxes represented by the paths. The check boxes, which are represented by a group of adjacent paths that differ in at most three elements, are then considered a check box group.

After the group identification, we analyze each group. For each group, we determine the *view* in the GUI to which the group belongs. Then, we determine $viewActionInstances(A', view)$, which are all lists of the subsequent action instances that were executed on this view. In each list in $viewActionInstances(A', view)$, we count for each check box belonging to the group, how many value selections referring to the check box are contained in the list. If this number is odd, we consider the check box as checked at the end of the list. If the number is even, the check box is considered unchecked. Then, we count the number of check boxes of the group that are checked at the end of the list. If this number is one, we consider only one of the check boxes to be checked during the execution of the list. As a list represents a usage of the respective view and, therefore, of the check box group, we consider that in this usage of the group, only one of the check boxes was checked. Finally, we count the number $n_{singleCheckboxChecked}$, which is the number of lists in $viewActionInstances(A', view)$, in which only one of the check boxes belonging to the group is considered checked at the end of the list. Based on this, we calculate the following ratio, being also the intensity of the smell:

$$R_{singleCheckboxChecked} = \frac{n_{singleCheckboxChecked}}{|viewActionInstances(A', view)|} \tag{4.5.12}$$

$R_{singleCheckboxChecked}$ focuses on a specific check box group and ranges between 0 and 1. It is 0 if the users never checked only one of the check boxes of the group when it was displayed. $R_{singleCheckboxChecked}$ is 1 if during 100% of times the check box group was displayed to the users, exactly one check box stayed checked at the end.

Possible Usability Critique Due to the direct relation to a check box group, this smell can provide concrete information, where the smell was observed. Currently, the smell does not consider check boxes already checked when a group is displayed. Hence, when analyzing the result, check boxes checked per default must be considered.

4.5.3.5. Misleading Click Cue

Foundation and Justification Users should be able to use a system with high efficiency. For this, they need to know which elements are helpful for fulfilling a task and which are not [94]. Typical actions that do not support the completion of a task are clicks on unclickable elements [8], e.g., pure text. In general, the visual design of a software should not mislead a user when fulfilling a task by including unclickable elements that look clickable [90].

Expected User Behavior If first-time users consider something clickable, they click on it if they think a click will give an advancement in fulfilling their tasks. Experienced users will not perform these clicks.

Expected Occurrence in Recorded Action Instances The recorded action instances will include instances of actions, which are clicks on unclickable elements like pure text or images without click handler.

Detection and Intensity For detecting this smell, we search the recorded data for instances of actions, in which unclickable GUI elements are clicked. As unclickable GUI element, we consider texts and images, as long as they are not children of a clickable GUI element. For each unclickable GUI element, we determine the number of times $n_{elementDisplayCount}$ the unclickable GUI element was displayed to the user. In addition, we count the number $n_{ineffectiveClicks}$ of performed clicks on the unclickable GUI element. Finally, we calculate the ratio between the two values, which is also the intensity of the smell:

$$R_{misleadingClickCue} = \frac{n_{ineffectiveClicks}}{n_{elementDisplayCount}} \tag{4.5.13}$$

The intensity $R_{misleadingClickCue}$ refers to a specific unclickable GUI element. It is 0 if the users never clicked an unclickable GUI element. It is 1, i.e., 100%, if, on average, the users tried to click an unclickable GUI element on every of its displays. The intensity can exceed 100% if, on average, there are more clicks on the unclickable GUI element than the number of times the element was displayed.

Possible Usability Critique Due to the reference to the unclickable GUI element, the smell can provide detailed information about which GUI element causes the smell and where it is located. It can further provide details on how often the element was clicked.

4.5.3.6. Required Text Field Focus

Foundation and Justification When opening a form, the first text field to be filled out should have the keyboard focus [91, 90]. This prevents users from performing the unnecessary action of changing the focus, e.g., through clicking into the text field.

Expected User Behavior If required, users will change the keyboard focus as their first action when a view is opened. Usually, this will be done through a mouse click.

Expected Occurrence in Recorded Action Instances The recorded action instances refer to the GUI element, on which they were observed. Through this, they also refer to the view, on which they were performed. Hence, the recorded action instances can be subdivided into the lists $viewActionInstances(A', view)$, each representing subsequent action instances that occurred in a specific $view$. The first action instance in such a list is then usually a click on the first text field, into which the users enter some data.

Detection and Intensity For the detection of this smell, we first determine all views. Then, for each view, we determine the actions $a_1 \dots a_n$ that were performed first when opening the view. This is done by calculating $\mathit{viewActionInstances}(A', \mathit{view})$ and determining the actions whose instances are the first elements in all lists in $\mathit{viewActionInstances}(A', \mathit{view})$. For each $a_i \in a_1 \dots a_n$, we determine $\mathit{firstInView}(a_i)$, which defines how often the action was executed first in the view. We ignore scrolling actions, as scrolling has no semantic meaning. This means, if scrolling was the first action when a view was opened, then we consider the second action instance as the first. Afterwards, we check if all $\mathit{firstInView}(a_i)$ for each $a_i \in a_1 \dots a_n$ follow an equal distribution. We check this by performing a chi-square test on all $\mathit{firstInView}(a_i)$, which compares their distribution with an equal distribution. We use a confidence level of 95% for this check. If this test observes, that the $\mathit{firstInView}(a_i)$ are not equally distributed, we determine the $a_i \in a_1 \dots a_n$ that was executed most often. If this action is a mouse click on a text field, we take $\mathit{firstInView}(a_i)$ and relate it to the number $|\mathit{viewActionInstances}(A', \mathit{view})|$ of all displays of the view to the users. As a result, we get the following ratio being also the intensity of the smell:

$$R_{\mathit{reqTextFieldFocus}} = \frac{\mathit{firstInView}(a_i)}{|\mathit{viewActionInstances}(A', \mathit{view})|} \tag{4.5.14}$$

The smells intensity varies between 0 and 1. It is 0 if an action was never the first to be executed in a view. It is 1, i.e., 100% if the action, for which the smell was detected, was always the first when the view was displayed.

Possible Usability Critique The smell refers to a view and a specific action, which is a click on a text field. Hence, it provides detailed information about the view of a software for which the smell was detected. Furthermore, it can inform about which text field is typically the one to be filled out first by the users.

Example Consider a view that was opened by users ten times. Once, the first action in the view was a click on a specific button. In the other nine times, the first action was a click on a specific text field. The smell detection will calculate the following intensity for the smell: $R_{\mathit{reqTextFieldFocus}} = \frac{9}{10}$, i.e., in 90% of all displays of the view, the first thing users did was setting the focus to the respective text field.

4.5.3.7. Good Defaults

Foundation and Justification When entering data, forms should be prefilled with default values where applicable [82, 83, 90]. These default values can be derived from statistics about entered data or from knowledge about the users. The defaults should match the data that most users are likely to enter.

Expected User Behavior If no default values are provided or if the default values do not match user requirements, then users will enter or change values in the corresponding GUI element if this is required. This means, any time the GUI element is displayed to a user, the user will change the value of the GUI element to something different from the default value. If the default value is already useful for the users, they will not change it.

Expected Occurrence in Recorded Action Instances The recorded action instances reference the GUI element, on which they were observed. Through this, they also refer to the view, on which they were performed. Hence, we can determine $viewActionInstances(A', view)$ being all lists of subsequent action instances that were performed in a specific *view*. If the value of a GUI element in a specific *view* is changed anytime it is displayed to a user, then each list in $viewActionInstances(A', view)$ contains a value change on this GUI element. Otherwise, the default value is used.

Detection and Intensity For the detection of this smell, we first determine all GUI elements and their corresponding view, that can be used to enter data. These are all GUI elements, into which a text input took place or on which a value choosing was performed. For each such GUI element, we determine $viewActionInstances(A', view)$ for the view to which the GUI element belongs. Then, for each list in $viewActionInstances(A', view)$, we determine the value that is chosen at the end of the list. If there is no value choosing in a specific list, we assume that the default value was chosen by the user. Through this, we determine all values chosen by the users on a specific GUI element, as well as the number $n_{valueChosen}$ of their choosing. Afterwards, we perform a chi-square test with a confidence level of 95% to check, whether the values are equally often chosen. If the values are not equally distributed, then we check which of the values is most often chosen. If this value is not the default value, we consider the smell present. The intensity of the smell is calculated by setting the $n_{valueChosen}$ of the value chosen most often into relation to the number of times the corresponding view was displayed. The resulting intensity is the following:

$$R_{badDefault} = \frac{n_{valueChosen}}{|viewActionInstances(A', view)|} \tag{4.5.15}$$

The smell's intensity varies between 0 and 1. It is 1, i.e., 100%, if a concrete value is always chosen by the users. The intensity cannot become 0. This is because it is specific to a concrete value and its calculation depends on the number of times this value was chosen. But, to be considered, a value is chosen at least once, which results in an intensity greater 0.

Possible Usability Critique Due to the relation to a specific GUI element, the smell can provide detailed information about where a default value should be set. In addition, it can provide potential good defaults, as it determines the values that are typically chosen by the users.

4.5.3.8. Unused GUI Elements

Foundation and Justification GUIs should not be crowded with too many elements that are not required by the user [94, 90]. If specific interaction elements are not used, this is an indicator that they are not required [45] and that they can be removed.

Expected User Behavior If a specific interaction element is not required, users will not use it or will use it only seldom.

Expected Occurrence in Recorded Action Instances When recording action instances, we record also the GUI model for a software. Unused GUI elements will show up in the GUI model, but will not be referenced by any action instance.

Detection and Intensity To detect this smell, we first determine all views utilized by users. Then, for each view, we determine the list of interaction elements belonging to the view. From this list, we remove all those interaction elements that are referenced by at least one action instance. If there remain unused interaction elements, we consider the smell present for the considered view. The intensity of the smell is then the ratio of unused to all interaction elements in the view:

$$R_{unusedGUIElements} = \frac{n_{unused}}{n_{all}} \tag{4.5.16}$$

The smells intensity varies between 0 and 1. It is 0 if there is no unused interaction element in the view. It is 1 if 100% of the interaction elements in the view are never used.

Possible Usability Critique As the smell has a relation to a specific view and also to the unused GUI elements, it can provide detailed information about where the smell was observed and which GUI elements are not required.

5. Implementation

For evaluating our approach for task tree generation and usability smell detection, we performed several case studies. In this chapter, we introduce the technical setup used in these case studies, which mainly consists of the tool suite for Automated Quality Engineering of Event-driven SofTware (AutoQUEST) [99]. The AutoQUEST environment supports the recording of events for different platforms and provides a generic infrastructure for processing them. Based on this, it is possible to generate usage profiles and usage-based test cases. In the following section, we describe how we utilized and extended AutoQUEST for our case studies. This covers the recording of action instances, the post-processing of events, the harmonization of GUI models, the generation of task trees, their transformation into other standards, and, finally, the derivations of grammatical structures from task trees for generating parsers for the language spoken between users and the software.

5.1. Recording of Action Instances

AutoQUEST provides functionalities for recording events, which are caused by action instances on different platforms [100]. This covers websites based on HTML and JavaScript, Java software using Swing and AWT for their GUIs, as well as software based on Microsoft Foundation Classes (MFC). For our case studies, we utilized the recording for websites and Java applications.

For recording website usage, AutoQUEST provides a monitoring server application with an Hypertext Transfer Protocol (HTTP) interface. Via this interface, it publishes a JavaScript. If a website shall be monitored, this script must be integrated into any page of the website, which nowadays can typically be configured in the CMS of a website. When a page is loaded by a web browser, the script registers with the event handling mechanism of JavaScript. It then handles any event caused by action instances and sends it to the monitoring server. Events are bundled to groups of ten events to prevent too many requests on the monitoring server. Furthermore, events are sent to the monitoring server if a timeout occurs or a user leaves a page. The monitoring server then stores the events in an XML format as files on the hard disk. An excerpt of an example log file for a login process on a website is shown in the following listing. It contains five recorded events. The first is the click on the user name field, which has the id *id1* as referred to by the target parameter. The second event is the pressing of the tabulator key to navigate to the password field. Afterwards follows the event indicating a change of the user name field to a new value. The fourth event is the changing of the password, which is then followed by a click on the login button. Each event has a time stamp as well as additional, event-specific information, e.g., coordinates of mouse clicks. The ids of the referred targets are not the DOM ids in a website but refer to detailed target information, which is also available in the log file.

```
<event type="onclick">
 <param name="timestamp" value="1404373276075"/>
 <param name="target" value="id1"/>
 <param name="Y" value="13"/>
 <param name="X" value="182"/>
</event>
<event type="onkeydown">
 <param name="timestamp" value="1404373278106"/>
 <param name="target" value="id2"/>
 <param name="key" value="9"/>
</event>
<event type="onchange">
 <param name="timestamp" value="1404373278111"/>
 <param name="target" value="id1"/>
 <param name="selectedValue" value="..."/>
</event>
<event type="onchange">
 <param name="timestamp" value="1404373280294"/>
 <param name="target" value="id3"/>
</event>
<event type="onclick">
 <param name="timestamp" value="1404373280332"/>
 <param name="target" value="id4"/>
 <param name="Y" value="20"/>
 <param name="X" value="26"/>
</event>
```

Listing 5.1: Excerpt of an AutoQUEST log file for website events.

The recording of events on websites can be as detailed as to contain confidential data. For example, in the above listing, the event for changing the user name field can contain a concrete user name. To prevent this, a website can be annotated using Cascading Style Sheets (CSS) classes on text fields, which indicate, that value changes on these text fields shall be recorded, but that the concrete values shall be omitted. As a result, the JavaScript does not record the values. An example of an event without recorded value is the changing of the contents of the password field in the above listing.

When the monitoring server receives recorded events, it needs to identify the client, of which the events were recorded. To allow this, the JavaScript determines an id for a client based on information about the used browser type, its version, and other information retrievable via JavaScript. This id is sent together with the events to the monitoring server, which stores the events into distinct log files, each belonging to a specific client id. Log files are closed after a timeout of 10 minutes if no user activity occurs. If the client with a specific id becomes active after the log file has been closed, a new log file is started to store the new events. Therefore, we record for any user session a separate log file without identifying the concrete user.

For monitoring Java Swing/AWT applications, AutoQUEST provides a library, that is used as a wrapper when starting the application to be monitored. This wrapper registers with the event chain of the GUI and, through this, records both, the GUI structure as well as the events caused by action instances. On each program start, the wrapper starts a new log file, which is closed when the program shuts down. There is no timeout that can elapse. Hence, also this monitor stores one log file per user session, whereas in this scenario, the user sessions can be much longer. The content of the log files is similar to the listing of the website monitor shown above and, hence, not described in more detail here.

For processing the log files, AutoQUEST provides tools that allow parsing the log files and applying different commands on the contained events. During this parsing, the platform specific events stored in the log file are mapped to the common event meta model described in Section 4.2. The same is done for the platform specific GUI structures. For example, a javax.swing.JButton from the Java Swing platform is mapped to a generic button representation in AutoQUEST. As a result, the recorded user sessions are represented as lists of generic events in the AutoQUEST transient memory. On these lists, we apply commands provided by AutoQUEST or added to AutoQUEST in the context of this thesis.

5.2. Post-Processing of Events

Although being transformed into a generic representation, events in AutoQUEST still have platform specific characteristics. For example, one difference between website and Java Swing/AWT based events is the level of detail. On Java Swing/AWT we record multiple key stroke events, whereas on websites we record one text change when a text is entered into a text field. Furthermore, events can have platform specific issues and misorderings that need to be corrected after having been parsed into AutoQUEST. Therefore, we implemented several AutoQUEST commands to overcome platform specific characteristics and problems as well as to harmonize event streams. Through this, we ensure, that after the application of these commands, each remaining event represents exactly one action instance. As the focus of this thesis is the processing of action instances and not of events, the details about the AutoQUEST commands for event post-processing and their effects can be found in Annex A. There, we also list, which commands were applied in the different case studies.

5.3. Harmonization of GUI models

A challenge, when working with GUI models, is the identification of the same GUI element in distinct log files. The reason is, that at runtime, there is no id for a GUI element, which is the same via many starts of the same software. For example, a specific button in a Java GUI has an id which may be different at the next start of the application. An exception are ids of HTML tags in DOMs of websites. If the CMS assigns such an id, it can be preserved via different user sessions or server restarts. But such ids are not always provided, which results in the same challenge as for desktop GUIs. To cope with this issue in the data of our case studies, we implemented both, an identification of the same GUI elements in different log files of the same software as well as a subsequent assignment of ids to HTML tags in DOMs of websites. For the first part, we use the path of a GUI element through the GUI model as identifying information, which helps us to create a GUI model as described in Section 4.3. The second part is done with an extension of the parsing process when loading log files of websites into AutoQUEST. Details for this process and its application in the case studies can be found in Annex B as this is not the focus of this thesis.

Websites and GUIs of desktop software significantly differ in their technical structure. For example, where a desktop software has one menu, a website normally has duplicates of a menu on each of its pages. For the user, these duplicates seem to be a single menu although technically they are not. This also means, that actions on the semantically same menu element on two different pages of a website are distinct. We call GUI elements, which are duplicated to many pages of a website *common page elements* [21]. To detect common page elements, and to consider them as equal, we extended AutoQUEST with a further command. This command works by checking the DOM paths of GUI elements on different pages. If the command finds paths on different pages, which are identical considering only DOM elements, it considers the respective GUI elements as common page elements. Through this, the targets referred to by the events parsed into AutoQUEST refer to one common page element, instead of referring to many duplicates. Hence, the events and the action instances they represent are also considered equal after the command is applied.

5.4. Generation of Task Trees

A challenge for the implementation was the detection of n-grams in task instance lists for performing the sequence detection described in Section 4.4.2. The reason is, that for large amounts of input data, many different n-grams exist, that may occur at many distinct positions in the task instance lists. To handle this, we applied a dedicated data structure and feature of AutoQUEST. The corresponding details are described in Annex C.

For the verification of our approach, we required visual representations of the generated task trees. One visualization comes with AutoQUEST and will be used in the case study descriptions. A further visualization used in the context of this thesis is based on ConcurTaskTrees (see Section 3). We implemented a transformation of our task trees to ConcurTaskTrees, which we describe in detail in Annex D.

5.5. Verification of the Task Tree Representativeness

The task trees generated by our approach need to be representative for the user behavior in general and not only for the recorded users. Otherwise, results of subsequent analyses using the generated task trees, such as our usability smell detection, cannot be generalized for unrecorded users. To validate the representativeness of the generated task trees, we perform several analyses in our case studies. In these analyses, we first generate task trees for one subset of recorded action instances of a website. Then we check if these tasks trees also describe action combinations in another subset of data recorded for the same website. To do this check, we implemented a verification approach that utilizes the grammatical nature of task trees when considering the "language spoken" between the user and the system (see Section 3.4). In this verification approach, we

1. transform a task tree generated on the first data set into a grammar of the language spoken by the user,
2. generate a parser based on the grammar, and
3. check for any n-gram of action instances in the second data set if the parser accepts the n-gram as a sentence of the parsed language.

Through this, we can determine how many action combinations of the second data set are executions of the task trees generated on the first data set. The higher the number of such action combinations, the more representative are the task trees. Below, we describe the transformation of a task tree into a grammar and how we generate and apply a parser based on them.

A task tree is already similar to a grammar. Therefore, it is possible to perform a transformation of a task tree into a grammar based on transformation rules. In our approach, the transformation rules are the following:

- The leaf nodes of the task tree become the terminal symbols in the grammar.
- The sequences, iterations, and selections of the task tree become non-terminals in the grammar.
- For an iteration i with child c, we generate two production rules of the form $i \rightarrow i\ c$ and $i \rightarrow c$.
- For a selection z with children $c_1 \dots c_n$, we create n production rules of the form $z \rightarrow c_i$.
- For a sequence s with children $c_1 \dots c_n$, we create one production rule of the form $s \rightarrow c_1 \dots c_n$.
- For a sequence s with children $c_1 \dots c_i \dots c_n$ where c_i can be optional, we generate a further production rule $s \rightarrow c_1 \dots c_{(i-1)} c_{(i+1)} \dots c_n$.

Through the last transformation rule, we ensure that also optionals are handled. An optional is not necessarily a direct child of a sequence s. Instead, it can be a child of a selection z, which itself is a child of s. Hence, in the last rule we do not check if the direct child of s is an optional, but if the child can be left out. A child can be left out, if a grandchild or a great-grandchild is an optional and there is no intermediate other sequence in the path from *s* to this optional. If a sequence has multiple optional children, then we create as many production rules as there are permutations for leaving out the optional children.

An example of a task tree resulting from our task tree generation is shown in Figure 5.1a. Its corresponding grammar is shown in Figure 5.1b. The terminal symbols and non-terminals in the grammar are represented and named the same way as the nodes in the task tree. The production rules for the same non-terminal are grouped together to better visualize, how many production rules are generated for a specific task. For example, for *Sequence 1*, two production rules are generated, as its second child can be left out.

Based on a grammar, we generate a parser for the corresponding task. A parser reads a given input and decides if the input is a valid sentence of the language that is defined by its grammar [101]. In our implementation, we use a simple LR parser (SLR) as its implementation is relatively simple [101]. The concrete implementation of such a parser is not important for this thesis and, hence, not described in this chapter. Please refer to the book of Aho et al. [101] for further details.

Using the parser generated for a specific task t, we check if a list of action instances contains an execution of t. For this, we first subdivide the list of action instances $a'_1 \dots a'_n$ into all permutations of possible n-grams $a'_i \dots a'_j \mid 1 \leq i < j \leq n$ of a minimum length of two. Then, we provide the parser with each of these n-grams and check if the parser returns an accept of the n-gram as a valid input. If we find a valid n-gram, we consider it as an execution of t. We call the action instances belonging to such an n-gram *matches*.

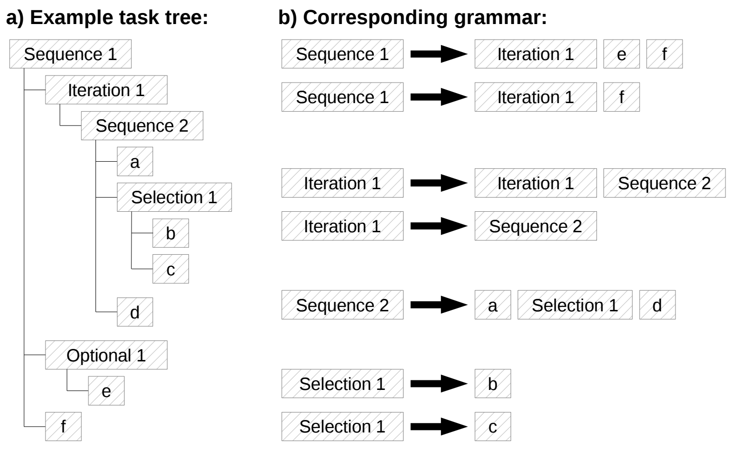

a) Example task tree:
b) Corresponding grammar:
Sequence 1
Iteration 1
Sequence 2
a
Selection 1
b
c
d
Optional 1
e
f
Sequence 1 → Iteration 1 e f
Sequence 1 → Iteration 1 f
Iteration 1 → Iteration 1 Sequence 2
Iteration 1 → Sequence 2
Sequence 2 → a Selection 1 d
Selection 1 → b
Selection 1 → c

Figure 5.1.: Example of a grammar transformed from a task tree generated by our approach.

For specific valid grammar constructs, it is not possible to generate an SLR parser. This is the downside that comes with the nature of these parsers [101]. Consider for example the following grammar:

$$\begin{aligned} A &\rightarrow BC \\ B &\rightarrow BD \\ B &\rightarrow D \\ D &\rightarrow ab \\ C &\rightarrow ac \end{aligned} \tag{5.5.1}$$

When reading an input, an SLR parser only considers the next symbol it reads. For the input *abac*, which is a valid input for the above grammar, the parser will parse over *ab*. For the next symbol in the input being *a*, the parser cannot decide if this is the first symbol of the symbol combination *ab* represented by the non-terminal *D* or of the symbol combination *ac* represented by the non-terminal *C*. But considering the grammar, both subsequent symbol combinations are valid. The parser could only decide this if it would consider one further symbol of the input, which is not done by SLR parsers.

The transformation of a task tree into a grammar may generate a grammar of the above or similar structures, for which no SLR parser can be generated. For example, in the above grammar, the non-terminal *A* can represent a sequence having the iteration *B* of sequence *D* as well as the sequence *C* as its children. In our case studies, we will show how often we cannot generate a parser for a task tree and, hence, how often we cannot check for the corresponding task if it is executed in a specific list of action instances. But, this is not a big disadvantage, as we generate parsers also for the children of a task. Hence, for the above example, we would have parsers for the sequences *D* and *C*. These parsers can also be used to identify executions of *D* and *C*, although the execution of *A* is not recognized.

6. Case Studies

In the context of this thesis, we performed three case studies to validate our approach and to answer our research questions. We describe these case studies in the upcoming sections. We start by introducing the commonalities between the case studies. Afterwards, we dedicate a respective section for each case study. In each of these sections, we provide a description as well as the results of the task tree generation and the usability evaluation. We finalize the chapter with a brief summary of additional experiments performed in the context of this thesis.

6.1. Case Study Setup

All case studies have a similar basic setup. With the different steps of this setup, we focused on answering our research questions, which we stated in the introduction of this thesis (see Section 1.2). We always started by recording users of a specific software. Then, we post-processed the recorded data as required, depending on the platform of the recorded software and other case-study-specific aspects. Through this, we filtered the recorded data for events representing only action instances. Afterwards, we compiled a GUI model and generated task trees, using the remaining action instances as input. With these initial steps, we answer the research questions RQ 1, RQ 1.1, and RQ 1.4. In two of the three case studies, we validated the representativeness of the task trees for recorded user behavior, which focuses on, and contributes to, answering the research questions RQ 1.2, RQ 1.3, and RQ 1.4. Afterwards, in all three case studies, we applied our detection of usability smells and analyzed the validity of the findings. This also includes a check if we detected different usability smells in merged and unmerged task trees. With these steps, we aimed at answering the research questions RQ 2, RQ 2.1, RQ 2.3, and RQ 2.3. Furthermore, we compared the findings with results of the application of established usability evaluation methods, which intended to answer the last research question RQ 2.4. The details for all steps which are common to the different case studies are described in the following.

6.1.1. Data Post-Processing and Task Tree Generation

In our case studies, we had to consider software-specific aspects. For example, events were not in their correct order or did not represent action instances. Hence, we performed the event post-processing as detailed in Annex A. Through this, the number of events recorded in a case study may have decreased, as some events were discarded. In addition, some sessions did not contain any event anymore. Therefore, we dropped these sessions. We list resulting number of sessions and events in the case study details.

The recorded websites only seldom provide ids for DOM elements. Hence, we performed a subsequent adding of these ids as described in Annex B. Corresponding to Section 4.4.2.3 and 4.4.3.2, the task tree generation and merging may fail. If we observed these problems in our case studies, we provide respective data in the detailed descriptions of the case studies.

6.1.2. Merging of Most Prominent Sequences

When merging sequences during the task tree generation, we did not perform a merge for all sequences detected in a case study. One reason for this is the large number of required sequence comparisons. In addition, the initial sequence and iteration detection finds all sequences that occur at least twice. It may also detect sequences, that cover only a few recorded action instances and which are, thus, less representative for the actual user behavior. Hence, we merged only a subset of the detected sequences that cover most of the recorded action instances. We call these sequences *most prominent* [23]. We consider all other detected sequences to represent noise in the data. We define the ratio of most prominent sequences to be 20% of all detected sequence. In our case studies, we show that this value is a good estimator for separating representative sequences from the others. In addition, we show that most prominent sequences also cover most of the recorded action instances and are, therefore, most representative for the user behavior.

As described in our previous work [23], when creating a subset of all sequences, as required for the most prominent sequences, a concrete percentage, e.g., 20%, can not be reached exactly. The reason is, that there may be multiple sequences with the same amount of covered action instances. Hence, also several combinations of sequences can be chosen to select a specific percentage, e.g., 20%, of sequences that cover most and also the same number of action instances. Therefore, we determine the most prominent sequences using the following process. We start by creating a sorted list of disjunctive sets of sequences $S_1 \dots S_n$. Each set contains all sequences that cover the same amount of action instances, i.e., $\forall s_i, s_j \in S_i : |a'(s_i)| = |a'(s_j)|$. The sorting order of $S_1 \dots S_n$ is given by the number of action instances that are covered by a sequence belonging to a set. This means, if S_i contains a sequence s_1, S_j contains a sequence s_2, and $|a'(s_1)| > |a'(s_2)|$, then S_i precedes S_j in $S_1 \dots S_n$. After this initial sorting, we determine the most prominent sequences. We start by analyzing S_1. If it contains less than 20% of all sequences, we join it with the next sequence set in the list being S_2. If the result still contains less than 20% of all sequences, we continue the joining with the respective next set in $S_1 \dots S_n$, until the result of the join contains at least 20% of all sequences. In the end, the join result contains the most prominent of all detected sequences. Through this approach, the most prominent sequences will usually be a little more than 20%. In the case studies, we will provide information about how many sequences are most prominent and how many of the recorded action instances they cover.

When merging the most prominent sequences, we use a value of 75% for the minimal similarity level sim_{min}. This means, that at most one in four elements of $L(t_1)$ and $L(t_2)$ are different.

6.1.3. Verification of the Task Tree Representativeness

In two case studies, we performed a validation of the representativeness of the task trees. For this, we randomly subdivide a data set A', which contains the post-processed action instances of a case study, into multiple subsets $A'_1 \dots A'_n \subset A'$ of almost equal size (i.e., $|A'_1| \approx |A'_2| \approx \dots \approx |A'_{n-1}| \approx |A'_n|$). Then, we generate task trees for one of the subsets $A'_i \in A'_1 \dots A'_n$, transform the detected sequences into grammars (see Section 5.5), generate parser for theses grammars, and check in the other subsets $A'_j \in A'_1 \dots A'_n \mid A'_j \neq A'_i$ how many subsequent action instances can be parsed with the parsers, i.e., how many matches these parsers have. Afterwards, we also check the amount of matches of the parsers in the full dataset A'. We only transform tasks of type sequence into grammars. The reason is that iterations, optionals, and selections are only variants of detected sequences, but sequences define the main structuring of a task.

In the case studies, we perform this validation with different subset sizes. These sizes depend on the size of A'. In addition, the number of subsets per size depends on the subset size and if the subsets are disjunctive to each other or not. The generated subsets are listed in Table 6.1. The first column is the subset size in % of $|A'|$. The second column contains the number of subsets created for the corresponding size, while the third column defines if the subsets are disjunctive.

Subsets are created based on user sessions as done in [23]. Through this, we ensure, that we do not split a data set in the middle of a user session or a task execution. The downside is, that user sessions have a certain size, i.e., number of action instances, and it may, therefore,

	Subset statistics		**Parsing attempts in**	
	Number	**Disjunctive**	**Subsets of same size**	**Respective full data set**
1%	50	true	20	20
2.5%	30	true	20	20
5%	20	true	15	15
10%	10	true	15	10
20%	5	true	15	5
30%	9	false	15	5
40%	6	false	15	5
50%	6	false	15	5

Table 6.1.: Sizes, numbers, and numbers of comparisons of subsets, which are created in the case studies for evaluating the task tree representativeness.

not be possible to exactly achieve an intended subset size, especially if subsets shall be disjunctive. In addition, it is not possible to consider all possible permutations of creating the subsets, as there are too many. Therefore, we created only one sample of possible session combinations to form subsets.

During the subset creation, the random assignment of sessions to a subset may result in the assignment of a session with many action instances to a subset that already has almost its intended size. Through this, a subset could exceed its intended size extensively. In such a case, we do not assign the randomly chosen session to the subset. Instead, we randomly search for another session that contains fewer action instances and still fits into the subset, so that the intended subset size is not exceeded. Through this, we ensure, that the subsets are of approximately the intended size. In the case studies, we will provide details on the number of action instances belonging to a subset on average.

After this preparation, we use the grammars to parse the action instances of subsets of the same size as well as of the respective full data set. The number of parsing attempts depends on the subset sizes and the number of subsets of the same size. Moreover, the task tree generation may fail for some subsets of a specific size and therefore, these subsets can not be used. Hence, we perform less parsing attempts than would be possible, considering the number of generated subsets and all permutations of their possible combinations. The concrete number of parsing attempts performed in our case studies is listed in the fourth and fifth column of Table 6.1.

For some subset sizes, we create rather few subsets to create them as disjunctive as possible. Nevertheless, they may all be needed to perform the verification against the respective full data set. For example, for the subsets of size 20%, all five subsets are used to parse the full data set. If for one of the subsets the task tree generation fails, then there are not sufficient subsets for a representative verification. In such a case, we performed a recreation of the subsets in the case studies, until for sufficient subsets of the respective size the task tree generation was successful. In the case studies, we mention the amount of times these recreations were required.

6.1.4. Usability Evaluation Analysis

In all three case studies, we applied our approach for usability smell detection on the full data sets. Regarding the task trees, we performed the analysis on both, the task trees before and after the merge. The reason is, that the task merging process has a high complexity and requires corresponding runtime. Therefore, we evaluate, if the usability smell detection provides better results on merged than on unmerged task trees, or if the task merging can be skipped. The distinction between unmerged and merged task trees is not required for the usability smells that are based on action instances. We provide the number of findings for the different smell types in the details of each case study.

A usability smell referring to a task may be detected for a task and also for one of its direct or indirect child tasks. This is because smells are detected on all tasks resulting from the task tree generation. We call these findings *duplicates*. We indicate in the case studies, how many of the findings for a specific smell type are duplicates.

After the usability smell detection, we performed a manual inspection of the findings to determine if they are true positives or not. For this, we analyzed the findings and checked, if the corresponding usability issues are present in the analyzed GUI. In the case studies, the number of findings for a smell can be very large. This makes a full manual inspection of all findings infeasible. Hence, we performed a preselection of the findings which we considered for manual inspection. In this preselection, we chose only those 30 findings of a specific smell type and data set, that covered the highest number of recorded action instances. If there were less than 30 findings per smell type and data set, we inspected all of them. If applicable, we mention in the case studies, how many of the preselected findings are duplicates.

In the case studies, we list for each usability smell, to which task group the corresponding findings belong. These task groups are logical groupings, i.e., a task group represents a set of tasks with a similar semantic, e.g., tasks representing a login process. If applicable, we provide distinct number for findings on unmerged and merged task trees for the task groups. Furthermore, we provide details on how many of the findings for a smell and a task group we consider true positives. If the findings for a smell in a case study belong to more than three task groups, we list the true positive findings in a corresponding table. Otherwise, we do a plain text description.

The smell descriptions do not include thresholds for the intensities of findings, which should be exceeded to consider a finding as true positive. One goal of our case studies, is to identify such thresholds. Therefore, we also analyze the intensities of the findings for the different usability smells and name values or value ranges that the intensities of true and false positive findings have.

In all case studies, we also compare the findings of our approach with the results of a user-oriented usability evaluation. For each case study, we describe the setup of this evaluation and mention the most important outcomes. Then, we compare how many usability issues found in the user-oriented usability evaluation were also identified, or at least indicated, by the findings for the usability smells. The number of test participants in these evaluations is in all three case studies large enough to get representative results (see Section 2.3).

6.1.5. Reasons for the Case Study Selection

In this thesis, we performed three case studies. All case studies serve different purposes when considering the validation of our approach. The first case study is a website where users mainly enter data. In contrast, the website of the second case study presents information to users. Finally, the third case study is a Java application with a combination of data input and presentation. Through this case study combination, we can validate our approach for different types of software (providing or consuming data) as well as for two separate platforms and interaction concepts (website and desktop application).

The website of the first case study is subdivided into two main subportals. One of them is a walk-up-and-use interface, which needs to be easy to use for first- and short-time users. In contrast, the other subportal is an expert system. Hence, the first case study covers these two different types of interfaces, which usually have different user requirements regarding learnability and efficiency. Furthermore, the first subportal is set up as a wizard, which also allows the evaluation of this kind of interface pattern. The second and third case study are software for medium regular use. For these software, users may recall elements of the interface, and their usage, during future usage sessions, but also need to explore some new parts on any use of the software.

6.2. Case Study 1: Master Application Portal

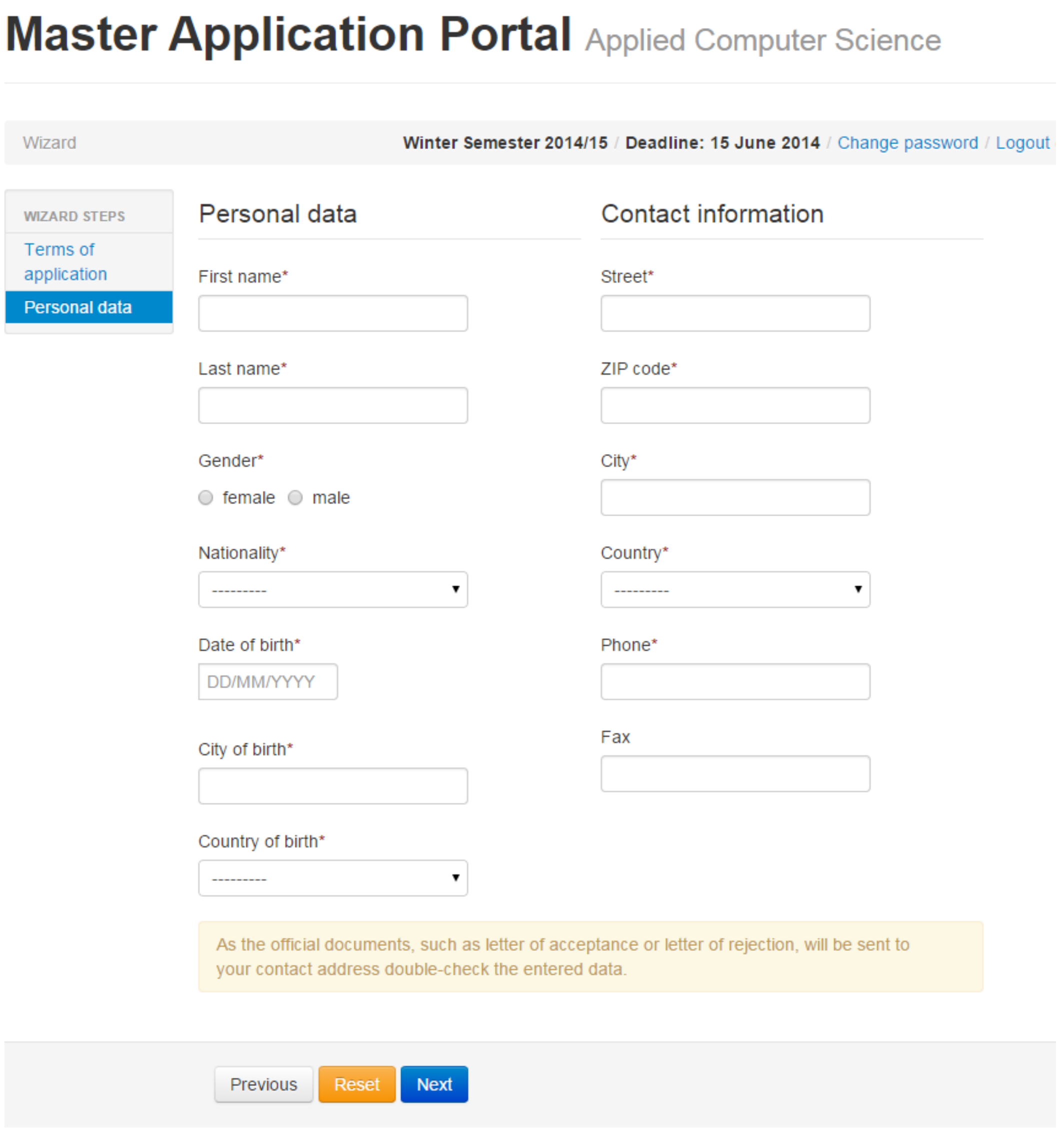
In the first case study of this thesis, we used our approach for analyzing an online application portal, i.e., a website. Via this portal, prospective students apply for the master studies in computer science at the University of Göttingen [102]. In this section, we first describe the analyzed software and provide several facts about the case study. Then, we list details and results of the task tree generation and the verification of the task tree representativeness. Finally, we present the findings of the usability smell detection and compare them with the results of a user-oriented usability evaluation of the same website.

6.2.1. Case Study Facts

The portal analyzed in this case study is subdivided into two separate subportals. The first, being the *applicants portal*, is used by prospective students to submit their application for the master studies in computer science. The second, called the *reviewer portal*, is used by the university employees to assess the applications and to decide if a student is accepted.

The applicants portal follows a wizard paradigm. This wizard guides the students through the different application steps, which cover entering of personal data, university degrees, and work experience. A screenshot of the wizard step for entering the personal data is shown in Figure 6.1. The students can step back and forth in the wizard, as required, using the navigation buttons at the bottom. Furthermore, the students can directly jump to an application step by using the wizard step navigation on the left. Depending on the information a student provides in a specific step, the subsequent steps of the wizard may be different. For example, if students select a specific type of how they learned English, they may be asked in a further step to provide detailed proof for this. At the end of the application process, students may be asked to upload several documents, for example a CV or a certificate of previous studies.

The reviewer portal focuses on the assessment of the students' applications. For this, it provides overview pages in a tabular fashion, allowing the reviewers to list all applicants or to filter, e.g., for those applications that were not yet assessed. In addition, the reviewer


Master Application Portal Applied Computer Science

Wizard | Winter Semester 2014/15 / Deadline: 15 June 2014 / Change password / Logout

WIZARD STEPS
Terms of application
Personal data

Personal data

First name*

Last name*

Gender*
○ female ○ male

Nationality*

Date of birth*
DD/MM/YYYY

City of birth*

Country of birth*

Contact information

Street*

ZIP code*

City*

Country*

Phone*

Fax

As the official documents, such as letter of acceptance or letter of rejection, will be sent to your contact address double-check the entered data.

Figure 6.1.: Screenshot of the applicants portal of the master application portal analyzed in the first case study.

portal contains pages displaying the details of a specific application. On these pages, the reviewers can make notes for individual aspects of an application and also provide a final decision if the application is to be accepted or not. Both subportals require a login. The login screen and other generic pages are shared between both subportals.

In this case studies, we recorded users over a period of 18 months, which resulted in about 1.4 million events and 19,000 user sessions with more than 2,750 distinct utilized client browsers. The concrete values are listed in the upper part of the first column of Table 6.2. We separated the recorded events depending on the subportal on which they were recorded. This resulted in 1,143,334 events and 17,725 sessions on the side of the applicants portal and 245,680 events and 1,547 sessions on the side of the reviewer portal. In addition, 7,149 recorded events belonged to 27 test sessions, in which both subportals were utilized. Further details for this subdivision of the data are listed in the second to fourth column of the upper part of Table 6.2. In the following, we refer to all recorded data in this case study as the *overall data set*. The separated data sets for the subportals are referred to as the *applicants portal data set* and the *reviewer portal data set*.

	Overall	**Reviewer portal**	**Applicants portal**	**Test sessions**
Recorded data				
Recording period	10/2013 - 02/2015 (18 months)	11/2013 - 02/2015 (17 months)	10/2013 - 02/2015 (18 months)	11/2013 - 11/2014 (13 months)
Events	1,396,163	245,680	1,143,334	7,149
Sessions	19,299	1,547	17,725	27
Distinct clients	2,757	162	2,574	21
Post-proc. data				
Events	807,654	147,458	656,100	4,096
Distinct actions	4,445	1,740	2,754	570
Sessions	16,266	1,428	14,811	27
Session length μ	49.7	103.3	44.3	151.7
Session length σ	129.5	323.0	88.9	287.1

Table 6.2.: Facts of the first case study including recorded and post-processed actions for the overall data set and the separate subportals.

The data was recorded in an anonymized fashion. This means, that the website was annotated with CSS classes as required by AutoQUEST to not record concrete values entered into specific text fields. Nevertheless, we still observed some personal data in unannotated text fields. Hence, we applied an algorithm on the data that pseudomized further text field entries. The pseudonymization was done using the SHA-512 hash algorithm. As a result, the recorded personal data was not human readable anymore, but identical values entered in different text fields remained identical.

After the pseudonymization of the data, we parsed it into AutoQUEST. Then, we let AutoQUEST detect common page elements. Finally, we applied several commands for post-processing the parsed events (see Annex A). The resulting number of events, which represent only action instances, and the number of remaining sessions are listed in the lower part of Table 6.2. Furthermore, this part of the table contains the average session length (μ), i.e., the average number of action instances in a session, as well as the corresponding standard deviation (σ). In addition, the table contains the number of remaining distinct actions that were executed. This number for the whole data set is not equal to the sum of distinct actions of the subportal data sets, as through shared pages some actions are shared by the subportals.

6.2.2. Task Tree Generation Results

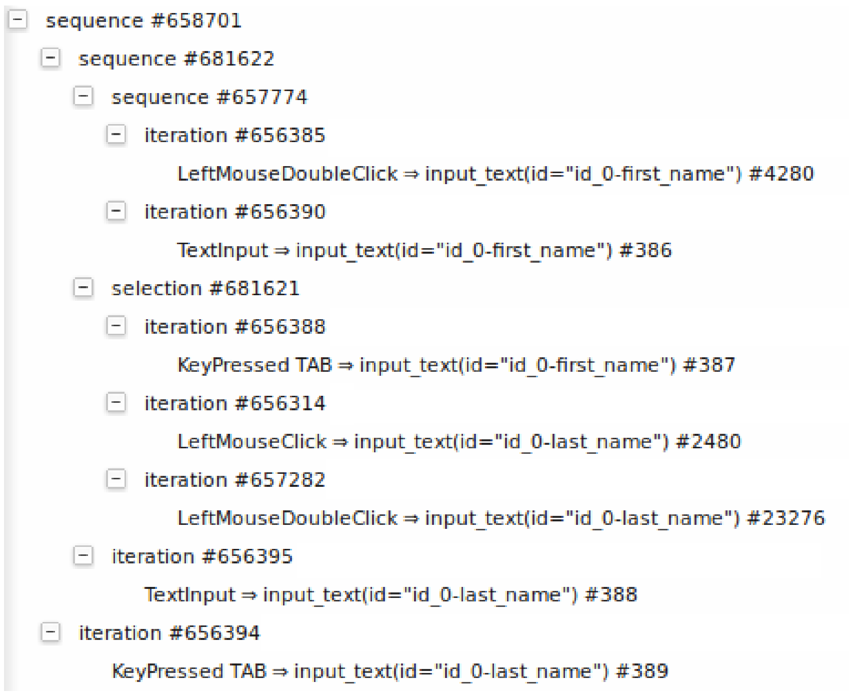
After post-processing the data, we generated task trees using our approach. We did both, generate task trees without and with subsequent merging. We did not generate task trees for the test sessions. The numbers of resulting tasks for the overall data set and for the subportal data sets are listed in Table 6.3. The upper part of the table shows the number of sequences and iterations before the merge as well as the concrete ratio of the most prominent sequences for the data sets. The middle part displays the number of sequences, iterations, optionals, and selections after the merge, and also the ratio of most prominent sequences. After the merge, the number of sequences is always smaller than the number of sequences before the merge. In contrast, the number of iterations increases. The lower part of the table contains three different ratios of recorded action instances. The first are the action instances covered by all sequences, the second are the action instances covered by the most prominent sequences before the merge, and the third are the action instances covered by the most prominent sequences after the merge. The ratio of action instances covered by the most prominent sequences after the merge is always higher than before the merge. The reason

	Overall	**Reviewer portal**	**Applicants portal**
Generated tasks			
Sequences	26,380	4,856	21,484
Iterations	2,827	698	2,145
Most prominent sequences	20,3%	20,3%	20,5%
After merge			
Sequences	25,158	4,608	20,508
Iterations	2,884	701	2,199
Selections	217	48	161
Optionals	168	31	119
Most prominent sequences	20,2%	20,4%	20,0%
Action instance coverage			
All sequences	95,3%	95,8%	95,1%
Most prominent	85,7%	84,6%	86,0%
Most prominent after merge	86,1%	85,2%	86,3%

Table 6.3.: Task trees generated in the first case study for the overall data set and the separate subportals.

is, that due to the merging of the most prominent sequences, their number decreases and, hence, also other sequences become part of the most prominent ones, which then may cover more action instances. When generating task trees for the three data sets in this case study, we did not have any task tree generation failures as described in Section 4.4.2.3 and 4.4.3.2.

An example of a task tree generated for the applicants portal for the form shown in Figure 6.1 is shown in Figure 6.2. The represented task shows the typical actions applicants take to enter their first and last name into the above form. They start by clicking on the first name field and entering their name. Then they move to the last name field by either using the tabulator key, a single mouse click, or a double mouse click. Afterwards, they enter their last name and move to the next form element by using the tabulator key.



```
sequence #658701
  sequence #681622
    sequence #657774
      iteration #656385
        LeftMouseDoubleClick ⇒ input_text(id="id_0-first_name") #4280
      iteration #656390
        TextInput ⇒ input_text(id="id_0-first_name") #386
    selection #681621
      iteration #656388
        KeyPressed TAB ⇒ input_text(id="id_0-first_name") #387
      iteration #656314
        LeftMouseClick ⇒ input_text(id="id_0-last_name") #2480
      iteration #657282
        LeftMouseDoubleClick ⇒ input_text(id="id_0-last_name") #23276
    iteration #656395
      TextInput ⇒ input_text(id="id_0-last_name") #388
  iteration #656394
    KeyPressed TAB ⇒ input_text(id="id_0-last_name") #389
```

Figure 6.2.: Example for a task tree generated for the form of the applicants portal shown in Figure 6.1 in the first case study.

6.2.3. Task Tree Representativeness

In this case study, we evaluated the representativeness of the generated task trees. For this, we performed the approach described in Section 6.1.3 for all three data sets of this case study. The details for the created subsets can be found in the Table 6.4. This table is subdivided into three subtables, each containing the subset information for either the overall data set (first subtable), the reviewer portal data set (second subtable), or the applicants portal data set (third subtable). The columns of each subtable contain the information for the subsets of a specific size. The first row of a subtable shows, how many subsets of a specific size are created. The second row contains the number of failed task tree generations (see Section 4.4.2.3). The third row lists the number of failed task mergings (see Section 4.4.3.2). The next two rows show the average (μ) number and standard deviation (σ) of events, i.e., action instances, that belong to a subset, as well as of the distinct actions in a subset. For example, the average number of events belonging to subsets of size 1% (first column) of the applicants portal data set (third subtable) is $6,561$, with a standard deviation of 0. The next two rows of a subtable show the average number and the standard deviation of the detected sequences and iterations for the subsets before the merging of sequences. The last four rows contain the average number and the standard deviation of the sequences, iterations, selections, and optionals for the subsets after the merging of sequences.

The number of failures for the task tree generation is 0 for the reviewer portal, 3 for the applicants portal and 6 for the overall data set. For the subset sizes from 10% to 50%, we create only a small number of subsets per size to create them as disjunctive as possible. To gain sufficient subsets, for which the task tree generation was successful, we did the following subset recreations (see Section 6.1.3):

- reviewer portal
  - recreation of the subsets of size 20% once
- applicants portal
  - recreation of the subsets of size 20% once
  - recreation of the subsets of size 50% once
- overall
  - recreation of the subsets of size 20% once
  - recreation of the subsets of size 40% once
  - recreation of the subsets of size 50% once

The number of recreations correlates with the remaining amount of task tree generation failures, which we did not solve through recreation. For example, for the overall data set, we performed most recreations of subsets and also have most of the remaining task tree generation failures (see Table 6.4). The remaining failures impose no problem, as there are still sufficient subsets of the different sizes for which the task tree generation and merging succeeded.

The intended average subset size is reached for the overall data set and the applicants portal data set with minimal deviations (standard deviation is at most 3 action instances for the overall data set). For the reviewer portal data set, the deviations are increasing with decreasing subset size. This is because the reviewer portal data set includes sessions that

Overall	**1%**		**2.5%**		**5%**		**10%**		**20%**		**30%**		**40%**		**50%**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Subset statistics																
No. of subsets	50		30		20		10		5		9		6		6	
Generation failures	0		0		1		0		0		3		0		0	
Merging failures	2		0		0		0		0		0		0		0	
Subset contents																
Events	8,076	0	20,191	0	40,383	3	80,765	1	161,531	2	242,296	0	323,061	0	403,827	0
Distinct actions	814	54	1,241	76	1,638	74	2,132	44	2,749	62	3,122	122	3,436	60	3,642	54
Generated tasks																
Sequences	634	24	1,375	47	2,430	60	4,228	52	7,342	56	10,080	112	12,726	60	15,208	86
Iterations	216	19	390	18	593	25	887	27	1,288	35	1,584	39	1,858	17	2,043	48
After merge																
Sequences	631	24	1,360	46	2,394	61	4,139	57	7,164	48	9,803	149	12,455	78	14,750	156
Iterations	217	19	391	18	594	25	890	27	1,295	37	1,599	38	1,867	15	2,067	48
Selections	2	1	5	2	8	2	15	2	25	8	38	16	27	27	58	34
Optionals	1	1	5	2	13	3	28	4	50	7	69	25	68	28	100	29

Reviewer portal	**1%**		**2.5%**		**5%**		**10%**		**20%**		**30%**		**40%**		**50%**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Subset statistics																
No. of subsets	50		30		20		10		5		9		6		6	
Generation failures	0		0		0		0		0		0		0		0	
Merging failures	0		0		0		0		0		0		0		0	
Subset contents																
Events	1,535	180	3,686	0	7,373	133	14,746	6	29,492	1	44,266	83	58,983	0	73,729	0
Distinct actions	220	30	334	44	468	55	687	51	914	59	1,114	47	1,237	15	1,352	43
Generated tasks																
Sequences	124	16	249	27	431	50	766	31	1,322	90	1,867	44	2,348	50	2,815	63
Iterations	40	7	64	10	101	16	164	12	263	11	341	19	416	9	471	15
After merge																
Sequences	124	16	248	27	426	50	751	31	1,283	90	1,809	44	2,264	49	2,699	76
Iterations	40	7	64	10	101	16	164	12	263	10	342	19	416	10	472	14
Selections	0	0	0	0	0	0	2	1	5	1	9	2	13	2	20	2
Optionals	0	0	1	1	3	2	7	2	14	2	19	3	24	3	33	2

Applicants portal	**1%**		**2.5%**		**5%**		**10%**		**20%**		**30%**		**40%**		**50%**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Subset statistics																
No. of subsets	50		30		20		10		5		9		6		6	
Generation failures	0		1		1		0		0		0		0		1	
Merging failures	0		0		0		0		0		0		0		0	
Subset contents																
Events	6,561	0	16,402	0	32,805	0	65,610	0	131,220	0	196,830	0	262,440	0	328,050	0
Distinct actions	646	42	935	46	1,187	42	1,497	47	1,845	22	2,061	21	2,201	30	2,344	14
Generated tasks																
Sequences	525	15	1,137	23	1,997	33	3,451	32	5,987	61	8,225	57	10,302	102	12,314	55
Iterations	189	14	333	19	500	18	728	24	1,035	19	1,253	29	1,434	14	1,583	21
After merge																
Sequences	523	15	1,124	22	1,967	33	3,380	30	5,830	57	8,012	95	10,066	181	11,898	36
Iterations	189	14	334	20	502	18	731	22	1,043	20	1,267	28	1,448	11	1,606	22
Selections	2	1	5	2	8	2	13	3	24	6	28	13	28	20	52	17
Optionals	1	1	4	2	10	3	21	5	41	4	53	14	50	21	85	13

Table 6.4.: Information about created subsets, generated task trees, and the comparisons done for the data sets of the first case study.

contain more than $1,800$ action instances. A session of this size can strongly influence the size of the subset to which it belongs, especially if the intended subset size is small.

The number of detected sequences and iterations is similar for different subsets of the same size. The standard deviation is rather small, in most cases, in comparison to the number of detected tasks. For example, for the applicants portal data set (third subtable) the standard deviation for the detected sequences on subsets of size 10% is 32, which is rather low in comparison to $3,451$ sequences detected on average for these subsets. In addition, the standard deviation decreases with increasing subset size. For example, for the reviewer portal data set (second subtable), the standard deviation for the number of sequences detected on the subsets of size 1% is 16, which is more than 13% of the average. In contrast, the standard deviation for the number of sequences detected on the subsets of size 50% is 63, which is only 2% of the average. This means, the larger a subset size, the more similar is the number of detected sequences and iterations for the different subsets. Similar effects are present for the task trees after the merge. Nonetheless, the standard deviations for the number of detected selections and optionals are relatively high. This is because, in comparison to the number of detected sequences and iterations for a subset, the number of selections and optionals is rather low.

After the subset creation, we visualized, how many recorded action instances are covered by the detected sequences before the merging of similar sequences. We did this for the subsets and also for the corresponding full data sets. The visualization for the reviewer portal data set is shown in Figure 6.3. The sequences form the x-axis. They are ordered by the number of action instances they cover. The first sequence on the x-axis is the one with the highest coverage. On the y-axis, we display the cumulative ratio of action instances that are covered by a specific sequence on the x-axis and all sequences left of it. The unit of both axes is percent. This means, that a specific ratio of the sequences that cover most action instances (specific point on the x-axis) covers a specific ratio of action instances (corresponding point on the y-axis). The plot contains several lines. The bold black line represents the sequences of the full reviewer portal data set. The thinner lines represent the sequences of subsets of sizes 2.5% (grey lines), 10% (red lines), and 40% (cyan lines). For each subset size, we plotted the lines for five subsets.

In Figure 6.3, we marked the amount of action instances (84.6%) that are covered by the most prominent sequences (20.3%) of the full reviewer portal data set before the merge. In addition, we marked the action instances covered by all detected sequences, which is 95.8%. The plot shows, that similar to the Pareto principle, already a small amount of sequences covers a large amount of action instances. This holds true also for the different subset sizes. Nevertheless, this effect decreases with a decreasing subset size. Furthermore, the plot shows that using a value of about 20% for separating most prominent from all sequences is well chosen. This is indicated by the gradient of the graph, which is high for the most prominent sequences and decreases for the other sequences. In addition, the plot shows, that the coverages for the different subset sizes are similar, as the five lines for each subset size are close to each other. Furthermore, the coverage increases with a higher subset

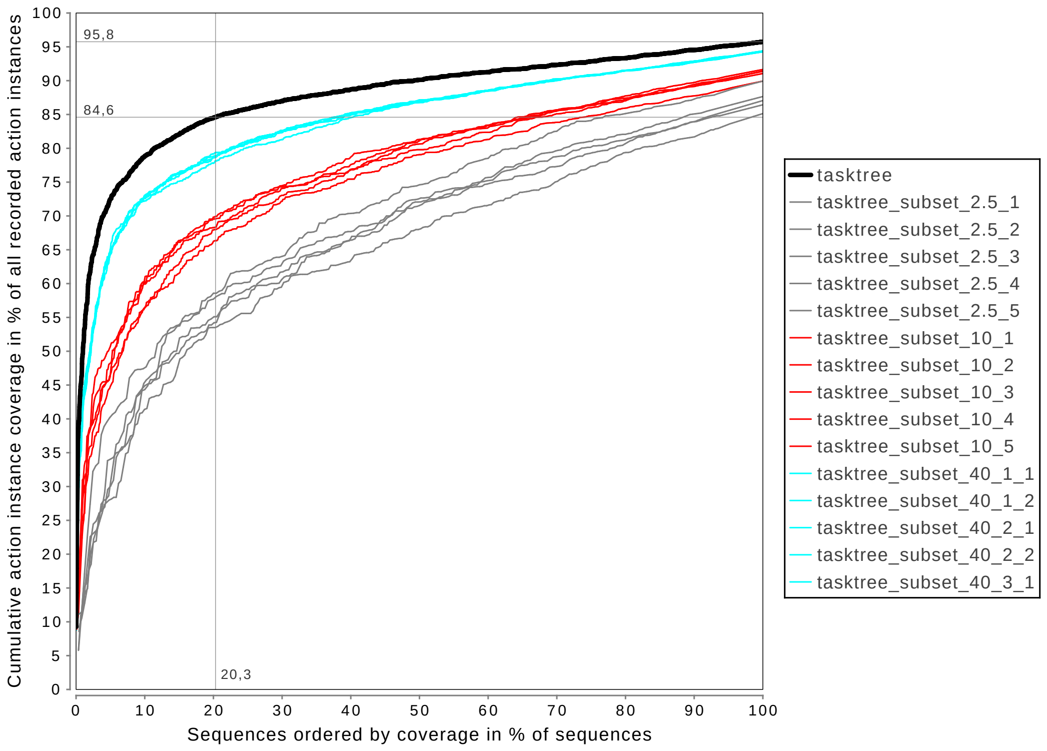


Figure 6.3.: Plot for the cumulative action instance coverage of the unmerged sequences of the reviewer portal data set (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

size. Finally, the larger the subset size, the more similar is the coverage distribution between the subsets of the same size. This corresponds to the decreasing standard deviation for the number of detected sequences listed in Table 6.4. The same type of plots for the applicants portal data set and the overall data set, as well as for the task trees after merging for all three data sets, are shown in Annex E.1. They show the same relation between the sequences and the covered action instances, as well as between the subsets and the respective full data set.

Based on the task trees of the subsets, we performed the evaluation of their representativeness as described in Section 6.1.3. As not all detected sequences can be transformed into a grammar, we only considered the sequences that could be transformed. Table 6.5 lists the average ratio (μ) and standard deviation (σ) of sequences of the different subsets, that were transformed into grammars and that were, hence, subsequently used in the analysis of their representativeness. In the table, we differentiate between the sequences before and after the merge. For all data sets, the ratio of transformed sequences decreases with increasing subset size. The differences between unmerged and merged sequences is at most 3.5% and increases with a higher subset size. As this increasing correlates to the number of selections and optionals generated through the merge, this may indicate, that the higher the number of selections and optionals, the less sequences can be transformed into grammars.

	Overall				**Reviewer portal**				**Applicants portal**			
	Unmerged		**Merged**		**Unmerged**		**Merged**		**Unmerged**		**Merged**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1%	96.4%	1.3	96.4%	1.4	90.9%	6.7	90.9%	6.6	96.9%	1.1	96.8%	1.2
2.5%	94.0%	2.0	93.9%	1.9	87.5%	7.6	87.5%	7.6	95.3%	0.8	95.1%	0.8
5%	92.3%	1.7	92.2%	1.5	82.0%	6.5	82.0%	6.1	94.5%	0.6	94.1%	0.7
10%	90.2%	1.8	89.5%	1.2	82.8%	5.2	83.0%	4.6	93.1%	0.7	92.4%	0.6
20%	90.6%	0.3	89.1%	1.2	82.2%	3.0	81.7%	2.8	92.1%	0.4	91.3%	0.9
30%	89.6%	0.2	87.2%	0.8	82.6%	0.0	81.1%	0.1	91.0%	0.3	88.8%	1.1
40%	88.3%	0.7	87.1%	1.8	81.8%	1.1	79.4%	1.2	90.7%	0.1	88.5%	1.6
50%	88.2%	0.2	86.6%	1.6	81.1%	1.4	77.9%	0.6	90.1%	0.3	86.6%	1.0

Table 6.5.: Average ratio of sequences that were transformed into grammars for checking their representativeness for other subsets of the same size as well as for the respective full data set in the first case study.

After the transformation into grammars, we generated parsers for each subset. Then we counted the number of matches that the parsers of a specific subset have in either other subsets of the same size or the respective full data sets. We did this several times for each subset size (see Table 6.1). Finally, we calculated for each subset size the average number of matches in other subsets of the same size and the respective full data set. We plot these results for unmerged task trees for the reviewer portal data set in Figure 6.4. The figure contains two bar charts. The left bar chart (Figure 6.4 a) represents the average matches in subsets of the same size. The matches are given in percent of action instances belonging to a subset. For each subset size (x-axis), there are two bars. The left bar (named c) shows the average number of action instances, which are covered by the transformed sequences (see Table 6.5) in the subset, from which they were generated. The right bar (named m) shows the average ratio of matches in another subset of the same size. The black part of a bar represents the coverage/matches of the most prominent sequences, the grey part the coverage/matches of the remaining sequences. For example, the sequences of subset size 2.5% (second bars in left chart) cover on average 82% of action instances in their own data set and match on average 62% of action instances in other subsets of the same size. The right bar chart (Figure 6.4 b) shows the average number of matches in the full reviewer portal data set. All bars, except the most right, represent the average number of matches by parsers, which were generated for the given subset sizes (x-axis). For example, for the sequences before the merge generated on subsets of size 10% (fourth bar in right chart), the average ratio of matches in the full reviewer portal data set is 81%. In this example, the most prominent sequences already cover 59%. The rightmost bar in this chart shows the ratio of action instances covered by all sequences detected on the full reviewer portal data set. We included this bar for easier comparison.

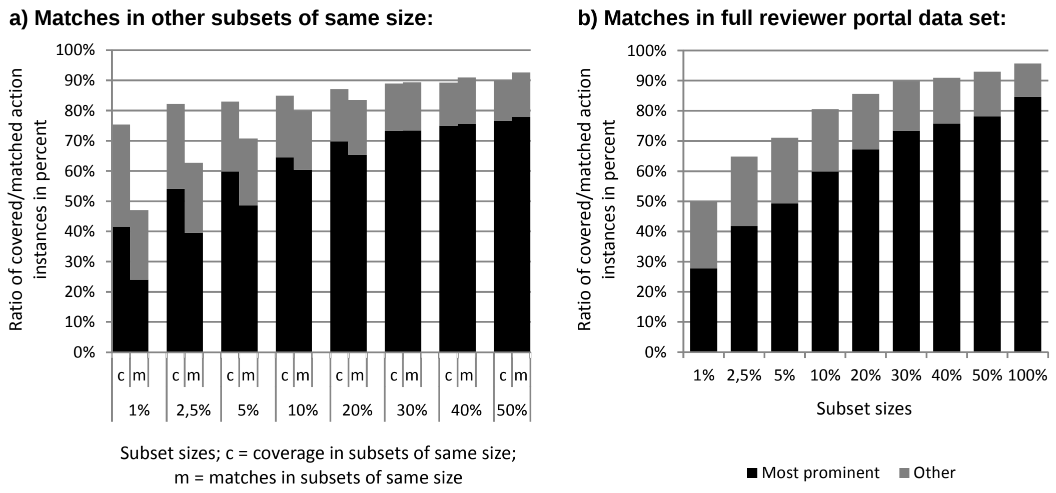


Figure 6.4.: Plot for the action instances matched by parsers, which were generated from unmerged task trees for a specific subset size of the reviewer portal data set in the first case study.

The plots show, that the average number of matches increases with a higher subset size. This holds true for the comparison with other subsets of the same size as well as with the full reviewer portal data set. In addition, the ratio of matches of the most prominent sequences increases even stronger. For example, the ratio of matches of sequences generated on subsets of size 1% is about 50% in the full reviewer portal data set. The most prominent sequences here match only 27%, which is about half of all matches. In contrast, the sequences generated for the subsets of size 50% match 92% of the action instances in the full reviewer portal data set. Here, the most prominent sequences match about 78% of all action instances, which is about 85% of all matched action instances. Furthermore, there is a break even point, after which sequences generated for subsets of a specific size match more action instances in other subsets of the same size than in the subset from which they were created. For example, the sequences generated on subsets with sizes smaller than 30% match less action instances in other subsets of the same size. In contrast, the sequences generated on subsets with sizes larger than 30% match more action instances in other subsets of the same size. The reason for this is, that the matching of sequences is much more flexible than their detection. For example, a subset may contain the action instances $\{a'b'c'\}$. Considering also other data in the subset, the sequence detection may find the sequence representing $\{ab\}$. Even if it would later detect another sequence representing $\{bc\}$, the action instance c' in the example would not be covered by a sequence, as the two preceding action instances are already covered by the sequence $\{ab\}$. But if we generate parsers for the sequences $\{ab\}$ and $\{bc\}$, then both parsers would match all action instances, where the parser for $\{ab\}$ would match the first two action instances and the parser for $\{bc\}$ would match the last two.

The corresponding plots for the merged task trees of the reviewer portal data set, as well as for unmerged and merged task trees for the applicants portal data set and the overall data set can be found in Appendix E.2. These plots show the same effects as seen in Figure 6.4.

6.2.4. Usability Evaluation Results

As the next step in this case study, we applied our usability smell detection on all data sets in this case study, except for the test sessions. The resulting number of findings for the different usability smells are shown in Table 6.6. The table consists of three subtables, each displaying the findings for a specific data set. The upper table shows the findings for the overall data set, the middle table the findings for the reviewer portal data set, and the lower table the findings for the applicants portal data set. Each subtable contains one row per smell type. In the upper part of a subtable, we list the findings for the smells with reference to the task trees, the lower part are the findings for the smells detected based on action instances. The usability smells are named in the left part of a subtable. For the upper part of a subtable, we provide the distinction between the smells detected without (center subtable part) and with (right subtable part) merged task trees. This distinction is not required, and, hence, not given, for the smells in the lower parts of the subtables, as these do not refer to the task trees. The number of findings for the smells is given in the first column of the respective table part. For example, in the reviewer portal data set (middle subtable) on merged task trees (right subtable part), we had $2,558$ findings for the smell "Required Inefficient Actions".

As indicated in Section 6.1.4, findings can be duplicates. We indicate for the findings in Table 6.6 the ratio of duplicates in percent in the second column of the respective subtable part. For example, of the above mentioned $2,558$ findings for the smell "Required Inefficient Actions", 30% were duplicates.

After the smell detection, we performed a manual inspection of the findings to determine if they are true positives or not. As due to their large number not all findings could be manually inspected (see Section 6.1.4), we list in the third column of the respective subtable part of Table 6.6, how many findings were manually inspected. If applicable, we mention in the fourth column the corresponding ratio of duplicates in percent. Finally, we mention in the fifth column of the respective table part, how many of the inspected findings we considered true positives given as concrete value and as ratio in percent. For example, of the above mentioned $2,558$ findings for the smell "Required Inefficient Actions" for the merged task trees of the reviewer portal data set, we manually inspected 30 findings, of which 43% were duplicates, and considered 24, i.e., 80%, of them as true positives. In the

Overall	**Before merge**					**After merge**				
	Findings		**Inspected**			**Findings**		**Inspected**		
	All	Dupl.	All	Dupl.	True positive	All	Dupl.	All	Dupl.	True positive
Based on task trees										
Important Tasks	352	57%	30	43%	30 (100%)	333	53%	30	30%	30 (100%)
Required Inefficient Actions	11,918	24%	30	27%	29 (97%)	11,530	24%	30	23%	29 (97%)
High GUI Element Distance	23,246	26%	30	40%	5 (17%)	21,972	25%	30	43%	6 (20%)
Missing Feedback	117	-	30	-	6 (20%)	117	-	30	-	6 (20%)
Required Input Method Change	12,193	25%	30	30%	0 (0%)	11,309	24%	30	27%	2 (7%)
Missing User Guidance	1	-	1	-	0 (0%)	1	-	1	-	0 (0%)
Based on action instances										
Required Text Format	217	-	30	-	4 (13%)					
Text Input Repetitions	646	-	30	-	3 (10%)					
Text Input Ratio	1	-	1	-	0 (0%)					
Single Checking of Checkboxes	21	-	21	-	0 (0%)					
Misleading Click Cue	173	-	30	-	16 (53%)					
Required Text Field Focus	25	-	25	-	17 (68%)					
Good Defaults	29	-	29	-	14 (48%)					
Unused GUI Elements	68	-	30	-	1 (3%)					

Reviewer portal	**Before merge**					**After merge**				
	Findings		**Inspected**			**Findings**		**Inspected**		
	All	Dupl.	All	Dupl.	True positive	All	Dupl.	All	Dupl.	True positive
Based on task trees										
Important Tasks	440	55%	30	70%	30 (100%)	415	54%	30	67%	30 (100%)
Required Inefficient Actions	2,717	31%	30	40%	29 (97%)	2,558	30%	30	43%	24 (80%)
High GUI Element Distance	4,284	36%	30	80%	0 (0%)	4,034	35%	30	73%	0 (0%)
Missing Feedback	71	-	30	-	2 (7%)	71	-	30	-	2 (7%)
Required Input Method Change	1,223	32%	30	40%	23 (77%)	1,134	31%	30	40%	23 (77%)
Missing User Guidance	1	-	1	-	0 (0%)	1	-	1	-	0 (0%)
Based on action instances										
Required Text Format	37	-	30	-	0 (0%)					
Text Input Repetitions	72	-	30	-	3 (10%)					
Text Input Ratio	1	-	1	-	0 (0%)					
Single Checking of Checkboxes	13	-	13	-	0 (0%)					
Misleading Click Cue	20	-	20	-	7 (35%)					
Required Text Field Focus	4	-	4	-	4 (100%)					
Good Defaults	7	-	7	-	3 (43%)					
Unused GUI Elements	21	-	21	-	5 (24%)					

Applicants portal	**Before merge**					**After merge**				
	Findings		**Inspected**			**Findings**		**Inspected**		
	All	Dupl.	All	Dupl.	True positive	All	Dupl.	All	Dupl.	True positive
Based on task trees										
Important Tasks	332	53%	30	40%	30 (100%)	324	50%	30	37%	30 (100%)
Required Inefficient Actions	9,145	22%	30	23%	29 (97%)	8,925	23%	30	27%	29 (97%)
High GUI Element Distance	18,907	24%	30	43%	6 (20%)	17,890	23%	30	33%	6 (20%)
Missing Feedback	50	-	30	-	5 (17%)	50	-	30	-	5 (17%)
Required Input Method Change	10,971	24%	30	30%	0 (0%)	10,165	23%	30	43%	0 (0%)
Missing User Guidance	1	-	1	-	0 (0%)	1	-	1	-	0 (0%)
Based on action instances										
Required Text Format	181	-	30	-	4 (13%)					
Text Input Repetitions	566	-	30	-	3 (10%)					
Text Input Ratio	1	-	1	-	0 (0%)					
Single Checking of Checkboxes	8	-	8	-	0 (0%)					
Misleading Click Cue	156	-	30	-	16 (53%)					
Required Text Field Focus	24	-	24	-	16 (67%)					
Good Defaults	27	-	27	-	12 (44%)					
Unused GUI Elements	51	-	30	-	0 (0%)					

Table 6.6.: Numbers of findings for the usability smells in the different data sets of the first case study, including the detection in unmerged and merged task trees.

following subsections, we provide details about the tasks and actions, for which smells were found, and why we considered the findings true positives or not. Per smell, we consider all findings for all data sets at once. This may sum up to at most 180 findings for a specific smell type that references task trees (30 findings for unmerged and 30 findings for merged task trees in three data sets) and 90 findings for usability smells referencing action instance.

6.2.4.1. Findings for Usability Smell: Important Tasks

For the usability smell "Important Tasks", we analyzed 30 findings for each of the data sets. For the reviewer portal data set, the findings focus on tasks for the visualization and assessment of details of a concrete application. For the applicants portal, the related tasks cover the uploading of files belonging to an application, the navigation of applicants in the wizard, the usage of a date chooser, as well as entering the degree gained for previous studies. Furthermore, there are tasks related to the user registration, login, and logout, which are shared between both subportals. The detailed numbers of findings for these tasks are contained in Table 6.7. On the left side of the table, we list the groups of tasks performed by the users separated into the reviewer portal and the applicants portal. The middle part of the table shows the findings for the task groups before (left two columns) and after the merge (right two columns) of detected tasks. The right part of the table provides the corresponding findings in the overall data set. A dash indicates, that for the given task group, there was no finding in the respective data set. Findings for the user registration, login, and logout task group are given separately for the subportal data sets (middle part of the table) and for the overall data set (last row, right table part). This is because for the overall data set, it cannot be distinguished anymore between the subportals for these tasks.

Findings for usability smell "Important Tasks"	**Corresponding data set**				**Overall data set**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Reviewer portal								
Assessment and commenting of application details	27	27	24	24	5	5	6	6
Visualization of application details	3	3	3	3	1	1	1	1
Combination of visualization/assessment of application details	-	-	3	3	-	-	-	-
Applicants portal								
Upload files at the end of application process	8	8	9	9	6	6	7	7
Navigation inside the wizard	4	4	5	5	3	3	4	4
Usage of date chooser	7	7	4	4	6	6	3	3
Entering of degree of previous studies	4	4	2	2	2	2	1	1
User registration/login/logout related tasks	7	7	10	10				
Shared pages								
User registration/login/logout related tasks					7	7	8	8

Table 6.7.: Numbers of detected "Important Tasks" usability smells in the different data sets of the first case study.

For the smell "Important Tasks", we considered all findings as true positives. The reason is, that this smell basically searches for sequences that are often executed and, hence, important for the users. For the manual inspection, we selected the smells referring to the sequences with the highest action instance coverage. Therefore, any of the referred sequences is important. The smell is not found for all sequences detected on a data set. This is caused by our implementation of the smell detection, which drops any smell with an intensity lower than 0.1. This means, that any smell referring to a sequence covering less than 0.1% of the recorded action instances are dropped. The intensities of the findings in all data sets varied from 0.5 to 9.2, i.e., the task with the highest action instance coverage covered 9.2% of action instances (here it was a task detected in the reviewer portal data set).

6.2.4.2. Findings for Usability Smell: Required Inefficient Actions

For the usability smell "Required Inefficient Actions", we inspected 30 findings for each of the data sets. The detailed numbers of the findings are listed in Table 6.8. The table is structured as the corresponding table for the previous smell.

Findings for usability smell "Required Inefficient Actions"	**Corresponding data set**				**Overall data set**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Reviewer portal								
Assessment and commenting of application details	15	15	13	12	5	5	3	3
Visualization of application details	8	8	6	5	2	2	2	2
Combination of visualization/assessment of application details	7	6	11	7	1	1	1	1
Applicants portal								
Upload files at the end of application process	16	16	12	12	13	13	12	12
Navigation inside the wizard	4	4	4	4	3	3	3	3
Insert/edit application details	3	3	3	3	2	2	3	3
Usage of date chooser	1	1	6	6	-	-	2	2
Submission of application	1	0	1	0	1	0	1	0
User registration/login/logout related tasks	5	5	4	4				
Shared pages								
User registration/login/logout related tasks					3	3	3	3

Table 6.8.: Numbers of detected "Required Inefficient Actions" usability smells in the different data sets of the first case study.

We considered 169 out of 180 findings for this smell as true positives. We did not consider findings as true positive if they included less than one inefficient in ten actions. This correlated to the intensities of the findings which was in most cases also below 10% for the false positives and above 10% for the true positive findings. As we considered only tasks in our inspection, and because the intensity calculation for this smell also takes into account the corresponding instances, there were few true positive findings with an intensity below 10% and false positives with an intensity above 10%. The intensities of the findings varied between 0.1% and 73%. The intensities of the false positive findings were at most 19.7%. In addition, we considered four findings with a high intensity as false positives which related to the submission of an application. Here, students get displayed all entered information at once for legal reasons and have to confirm them. This is too much to fit on the screen. Hence, scrolling is required but can also not be prevented.

We had findings, where the tasks contained only few actions, of which the first or the last were inefficient actions. We considered it hard for these tasks to decide, whether the smells are true positives or not. We counted them as true positives, anyway, as these tasks had a relatively high action instance coverage.

We observed, that due to the merging of tasks, the resulting tasks contained optional inefficient actions and had a higher action instance coverage (the sum of the action instances covered by the merged tasks). The intensities of corresponding findings were rather low because, although the model contained inefficient actions, the instances showed that these were executed rather seldom. These were also the task, were the model contained more than one inefficient in ten actions but the intensities were below 10%. This shows, that the intensity calculation correlates with the structure of the tasks.

One of the major findings based on this smell is, that users of the applicants portal have to do a lot of scrolling when uploading files at the end of the application process. On the corresponding web page, users are presented with an overview of the whole application data as a large table. This table includes interaction elements to upload files such as a CV. After each file upload, the page is reloaded and users have to scroll down to upload the next file. The corresponding detected smells had a relatively high intensity.

6.2.4.3. Findings for Usability Smell: High GUI Element Distance

For the usability smell "High GUI Element Distance", we analyzed 30 findings for each of the data sets. The detailed numbers of the findings are listed in Table 6.9. We considered only 23 out of 180 findings as true positives. The reason is, that many smells referenced tasks for which the required GUI elements are located close to each other in the corresponding views. This correlated with the intensities of the findings. These were below 0.5 for the false positives, which indicates that the subsequently required GUI elements are located in the same view. One example for a false positive finding for the reviewer portal was, that when assessing the details for an application the corresponding links, check boxes, and text fields are close to each other on the same page. The only situation, where the findings were helpful, was for the uploading of files at the end of the application process. The GUI elements for performing the upload are not located as closely to each other as would best support the process. Therefore, we considered the corresponding findings as true positives, although their intensities were rather low and varied between 0.217 and 0.275.

What we observed is, that overlays may be misinterpreted regarding their location. For example, in the applicants portal, there are many text fields into which the users have to enter a date. When clicking on the text field, a date chooser opens as overlay of the website

Findings for usability smell "High GUI Element Distance"	**Corresponding data set**				**Overall data set**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Reviewer portal								
Assessment and commenting of application details	26	0	24	0	6	0	8	0
Combination of visualization/assessment of application details	4	0	6	0	2	0	1	0
Applicants portal								
Upload files at the end of application process	6	6	6	6	5	5	6	6
Navigation inside the wizard	1	0	-	-	-	-	-	-
Insert/edit application details	7	0	6	0	6	0	4	0
Usage of date chooser	10	0	9	0	7	0	6	0
User registration/login/logout related tasks	6	0	9	0				
Shared pages								
User registration/login/logout related tasks					4	0	5	0

Table 6.9.: Numbers of detected "High GUI Element Distance" usability smells in the different data sets of the first case study.

close to the text field. But regarding the GUI model, the date chooser has a relatively high distance. Hence, we considered corresponding findings as false positives.

6.2.4.4. Findings for Usability Smell: Missing Feedback

For the usability smell "Missing Feedback", we analyzed 30 findings for each of the data sets. The detailed numbers of the findings are listed in Table 6.10. The findings were the same for merged and unmerged task trees, as the considered tasks (iterations of actions) are not affected by the merging process. We considered only 24 out of 180 findings as true positives. The reasons for this are manifold. Many findings referenced iterations with only few repeated clicks in their instances. This correlated with the rather low intensities of below 266ms[3]. For example, for the reviewer portal data set, 28 out of the 30 inspected findings for the unmerged task trees indicated more an occasional reuse of an interaction element than a required one, which resulted in an intensity lower than 25ms. In contrast, the other two findings were considered as true positive. The first of these two findings with an intensity of 1032ms was given for the switching between tabs of a tabular panel providing overviews of applications. When the users switch between these tabs, the content of the selected tab is loaded in the background without a page reload. This takes some time in the case many applications are stored in the system, but the system does not provide any user feedback while loading. Hence, the users click the tab again, as it appears, that the first click did not have an effect.

[3]The intensities have a unit of milliseconds, which is due to the way they are calculated (see Section 4.5.2.4). But they do not represent a single repetition of a click. Instead they are the result of a calculation, that considers the number of repetitions, the repetition time, and all instances of the corresponding iteration.

Findings for usability smell "Missing Feedback"	**Corresponding data set**				**Overall data set**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Reviewer portal								
Assessment and commenting of application details	18	0	18	0	9	0	9	0
Visualization of application details	11	2	11	2	3	1	3	1
User registration/login/logout related tasks	1	0	1	0				
Applicants portal								
Upload files at the end of application process	3	1	3	1	2	1	2	1
Navigation inside the wizard	12	1	12	1	6	1	6	1
Reset form for application details	1	1	1	1	1	1	1	1
Insert/edit application details	3	0	3	0	1	0	1	0
User registration/login/logout related tasks	11	2	11	2				
Shared pages								
User registration/login/logout related tasks					8	2	8	2

Table 6.10.: Numbers of detected "Missing Feedback" usability smells in the different data sets of the first case study.

For the applicants portal, some findings were considered as false positives, as they were found for common page elements reused after a page load. For example, the next button available on any wizard page is a common page element. Some users navigated through the wizard by repeatedly clicking the next button in the wizard, what caused an appropriate false positive finding for the smell. In contrast, we found a true positive for the usage of the reset button available on any wizard page, which can be used to clear the current form. If the form is already empty, there is no visual feedback for the users for a click on the reset button.

6.2.4.5. Findings for Usability Smell: Required Input Method Change

For the usability smell "Required Input Method Change", we analyzed 30 findings for each of the data sets. The detailed numbers of the findings are listed in Table 6.11. We considered only 48 out of 180 findings as true positives. These findings all referred to the assessment and commenting of application details in the reviewer portal. Here, the reviewers perform several clicks to open a text area, in which they type comments on different aspects of an application using the keyboard. To submit a comment, the reviewers have to use the mouse again. Pressing the enter key does not submit the comment, which should be possible. The intensities of the findings varied between 0.5% and 96.4%.

The findings for the usage of the date chooser in the applicants portal are false positives. Here the problem occurred, that after the usage of the date chooser the final event was a text input in the corresponding text field. This event does not represent an action instance. Instead, it is generated in the background, because the selected date is automatically entered into the text field. The text input in the text field is considered a keyboard usage, following

Findings for usability smell "Required Input Method Change"	**Corresponding data set**				**Overall data set**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Reviewer portal								
Assessment and commenting of application details	26	23	28	23	-	-	2	2
Visualization of application details	1	0	1	0	-	-	-	-
User registration/login/logout related tasks	3	0	1	0				
Applicants portal								
Upload files at the end of application process	6	0	7	0	6	0	6	0
Insert/edit application details	4	0	3	0	4	0	2	0
Usage of date chooser	11	0	9	0	11	0	9	0
User registration/login/logout related tasks	9	0	11	0				
Shared pages								
User registration/login/logout related tasks					9	0	11	0

Table 6.11.: Numbers of detected "Required Input Method Change" usability smells in the different data sets of the first case study.

the mouse usage for the date selection. The same effect happened for the uploading of files, where the file selection is done by using the mouse, which is succeeded by an automatic text input into a text field.

A final issue causing false positives is, that users utilized the mouse to navigate between text fields. Hence, there were tasks that were combinations of clicking into a text field and entering some text. The smell did not check if there are other tasks occurring more often in which the tab key was used for navigating to the text field instead of using the mouse. The intensities of all false positive findings varied between 0.3% and 98,3%.

6.2.4.6. Findings for Usability Smell: Missing User Guidance

Due to its nature, the usability smell "Missing User Guidance" returns one finding per data set as long as no threshold has to be exceeded regarding the intensity. The intensity of the smell varied between 2.38 for the reviewer portal data set and 2.78 for the applicants portal data set. The intensities were the same between unmerged and merged task trees, which was also expected due to the smell's nature. The intensities are rather good, indicating that on average 10 subsequently executed actions are described by 2.38, respectively, 2.78 tasks. These values are considered rather good and, hence, the smell findings are all false positives.

6.2.4.7. Findings for Usability Smell: Required Text Format

For the usability smell "Required Text Format", we analyzed 30 findings for each of the data sets. We considered only 8 of the 90 findings as true positives. There are several reasons for this. For example, in 80 out of 90 findings, only a few users entered special characters into a specific text field. These findings also had a rather low intensity of less than 7.5% and would not occur if there was a threshold for the smell's intensity. Furthermore, there were findings for date (16) and file name (10) fields. Usually, these are not filled manually by the users, but automatically after the users used the date or file chooser. Hence, we did not consider them as true positive. Finally, there were 10 findings for fields for entering e-mail addresses or phone numbers. For the first, the format of an e-mail address is known by users today and should not be a problem for them. Phone numbers may be more problematic, but the portal does not check the numbers for a specific format and, hence, does not require the special signs. These findings contradict with our attempts to pseudomize personal data in this case study. A subsequent check showed, that even after pseudonymization, there were a few e-mail addresses and telephone numbers remaining in the data. The intensities of all false positives varied between 0.2% and 16.1%.

The eight true positive findings were all related to text fields for entering a degree of previous studies in the applicants portal. Their intensities varied between 5.0% and 28.6%. The degree is either without decimals, then no format is required, or it is with decimals, then the decimal numbers must be separated using a dot. The smell detection found the required format and announced it correctly.

6.2.4.8. Findings for Usability Smell: Text Input Repetitions

For the usability smell "Text Input Repetitions", we also analyzed 30 findings for each of the data sets and considered 9 of the 90 findings as true positives. For the reviewer portal, 27 out of the 30 findings were occasional text repetitions and, therefore, not considered as true positives. Their intensities were below 6% and correlated with our decision. The remaining smells referred to text fields for assessing the grade of an applicant. As grades for different countries have different ranges, there is a calculator for reviewers that allows to transfer grades into a harmonized grading system. The entries of this calculator were often transferred manually to the assessment details of an application, although this could be automated. Hence, these findings were considered true positives. With a range of 13% to 35%, the intensities for these true positives where higher than those of the false positives.

Also for the applicants portal and the overall data set, 27 out of 30 findings were occasional text repetitions letting us consider them as false positives, too. Their intensities were rather low, i.e., less than 10%. Two of the other findings related to a combination of date fields into which applicants enter information about their former studies. Here, they have to enter the date of when their studies finished and also when they were awarded with their degree. In 25% of times, applicants entered the same dates here. In addition, when applicants registered as a new user for the applicants portal, they used the same user name twice, first for registration and then for login[4]. Both situations could be automated. Hence, we considered the corresponding findings as true positives.

[4]This finding is based on the pseudomized data. Although being unreadable, the entered texts where encrypted the same way and, hence, identical.

6.2.4.9. Findings for Usability Smell: Text Input Ratio

Due to its nature, the usability smell "Text Input Ratio" returns one finding per data set as long as no intensity threshold has to be exceeded. The intensity of the smell varied between 4% for the reviewer portal data set and 10% for the applicants portal data set. The intensities were rather low, as inefficient actions are included in the calculation. The intensities correlate with the subportals usage. Where on the applicants portal, students have to enter a lot of data into text fields, the reviewers interact more with the mouse to do the applications assessment. Anyway, we considered the findings false positives as both subportals only use text fields where really required.

6.2.4.10. Findings for Usability Smell: Single Checking of Checkboxes

For this smell, there were only 13 findings for the reviewer portal and 8 findings for the applicants portal. The findings for the overall data set are the joined findings of both subportals. We considered none of the findings as true positives, as all check box groups really need to be check boxes. For example, at the end of the application process, the applicants can provide information about how they got to know about the study program. Here, they can select different sources of information, of which several can be possible at once. In addition, although this finding had the highest intensity of 58.6%, the intensities of all other findings were below 25%, which indicates that relatively often not exactly one check box is selected at the end of displaying a view.

6.2.4.11. Findings for Usability Smell: Misleading Click Cue

For this smell, there were 20 findings for the reviewer portal, 156 for the applicants portal, and 173 for the overall dataset. We inspected all findings for the reviewer portal, as well as 30 findings for the other two data sets. For the reviewer portal, we considered 7 findings as true positives. Their intensities varied between 0.2% and 5.1%. The other findings covered less than 10 action instances and were, thus, not representative in our opinion. The 7 true positives referred to a disabled user registration button, unclickable headlines (2 findings), and unclickable reviewer comments (4 findings). The findings for the user registration button show, that the difference between enabled and disabled for the button may not be obvious for all users. In addition, there is no direct feedback, why a user registration is not possible at present. The findings for the headlines focus on the major headline "Master Application Portal" available on any page of the portal, as well as the subheadline following it (see Figure 6.1). These headlines have no function when clicking on them, but should lead the users back to an initial page instead. This initial page could either be an overview page if users are logged in or the login page if users are not logged in. The unclickable reviewer comments referenced by the remaining 4 true positives are displayed, e.g., when showing details of an application. A click on the comments should lead the users to the change log of the application assessment.

The inspected findings for the applicants portal data set and the overall data set were the same. We assessed 16 out of the 30 findings as true positives. Their intensities varied between 0.3% and 63.1%. One finding referenced the disabled user registration button and four findings the unclickable headlines. In addition, there were 5 findings for the labels in front of two date fields. These fields are the start and the end date of the applicants previous studies. The labels here look different from other labels in the wizard and are attached to the text field. Hence, they suggest to be clickable although they are not. Therefore, we considered these findings as true positives. The other findings were rather distributed and considered smaller issues, like unclickable names of uploaded files. The 14 findings we considered as false positives were all related to the date chooser. This component contains some labels which are clicked by users to navigate, e.g., between the previous and next years. Although these labels are no links or buttons, they can be clicked. A Javascript handles the clicks and takes the required actions. Here, the smell detection was wrong, because the Javascript makes normally unclickable elements clickable. The intensities of the false positives findings were between 1.4% and 30.7%.

6.2.4.12. Findings for Usability Smell: Required Text Field Focus

For the smell "Required Text Field Focus", our detection resulted in 53 findings in all three data sets (24 applicants portal, 4 reviewer portal, 24 overall data set), of which we considered 37 as true positives. The intensities of the true positive findings were between 28% and 100%, the intensities of the false positive findings between 11% and 52%. All the true positives related to different views, where there was no default cursor positioning when a view was opened. An example is the login page. 8 out of the 16 findings that we considered as false positives were for views in the applicants portal. These views contained forms where the first form element was not a text field but a combo box. These views also did not have a focus on the first combo box after loading. Hence, these findings were also indirect hints for usability issues. The remaining 8 findings did not reference views with a missing text field focus and were, therefore, considered as false positives. Their intensity was below 15% correlating with our assessment.

6.2.4.13. Findings for Usability Smell: Good Defaults

Of the 63 findings for the smell "Good Defaults" (27 applicants portal, 7 reviewer portal, 29 overall data set), we considered 29 as true positives. These findings showed mainly the standard selections applicants took when filling out the application wizard. For example, our data showed, that most applicants selected the gender male or that their previous studies were full time studies. But some of these findings also have to be considered with caution. For example, although selecting the gender male would be a good default from the usability point of view, it is not from an ethical perspective considering equal opportunities. 24 of the false positives were related to unrecorded personal data and passwords of the applicants.

This is caused by the fact, that if the contents of a specific text field are not recorded by our approach, the stored data is an empty string for each text input. Hence, our evaluation assesses the empty string as a "good default". But this is no helpful finding. The remaining four false positives were related to groups of check boxes and showed the typical selections users took. But as with other findings, these defaults should not always be preselected. For example, one of the findings referred to the information about how the applicants got to know about the applicants portal. This questionnaire providing statistics should not be biased by preselected check boxes.

6.2.4.14. Findings for Usability Smell: Unused GUI Elements

The findings for the smell "Unused GUI Elements" were the hardest to be analyzed. The findings were polluted with unused GUI elements, which did not belong to the portal, but were automatically added by browser plugins at client side. In addition, there were findings for GUI elements, whose position in the GUI model changed once, because of an additional message displayed to a user. Finally, there were GUI elements that had DOM ids, which were session specific. If these elements were not used in the session to which they belonged, they were identified as never used also by other users. Due to these issues, the intensities of all findings for this smell were unexpectedly high and of not much worth.

We considered all of the 30 findings for the applicants portal, which were the same for the overall dataset, as false positives. They only referenced noisy GUI elements that do not belong to the applicants portal. For the reviewer portal, there were 5 out of 21 findings that we considered as true positives. Two of them were related to filters on overview pages. For example, a reviewer can list all applications that are yet incomplete. This list can then be filtered further, e.g., for applications from students coming from a specific university. The list of available filters is rather long, and not all filters were used by the reviewers. Hence, these findings were considered as true positives. Perhaps, the number of available filters could be reduced, or not all filters need to be available on all overview pages. Two further findings were related to adding a new reviewer account to the system. During this process, a reviewer can be assigned a fax number. But as indicated by the findings, this was never done. Hence, these findings are true positives. Finally, the reviewer portal offers a functionality to automatically assign a list of available reviewers to applications. This is done to have an even distribution regarding the number of applications the reviewers shall assess. This function was never used and the corresponding finding was, hence, considered as true positive.

6.2.5. Result Validation: Application of a User-oriented Usability Test

To validate the results of the usability smell detection, we performed a user-oriented usability evaluation of the applicants portal. In this evaluation, we let ten test participants do a complete application process. The test participants were bachelor students at our university. If they wanted to advance to the master studies, they would need to use the applicants portal to apply for it. Hence, they were representative prospective users for the portal. Nonetheless, they did not represent all user groups of the portal, as there are usually many applications of foreigners. One of the test participants was female, the others male. They all considered their English skills to be between medium to very good. Two of the test participants saw the system before, as they were in the process of preparing their application for the master studies. But none had finalized the application.

All test participants performed a full application process, which we subdivided into the five logical units shown in the left column of Table 6.12. On average, performing a full application took the test participants 37.4 minutes with a standard deviation of 5.8. When the test participants had to upload files, e.g., a CV, we asked them to upload dummy files, but to tell us if they would know, which file would be required in a real life scenario. During the application process, we observed the test participants and made notes about the problems they had. Furthermore, we asked them to do Thinking Aloud (see Section 2.3). Afterwards, we counted the problems the test participants had and considered every problem that occurred for at least two test participants as usability issue. Every problem that occurred for five or more test participants was considered severe. The number of found usability issues using this process is given in the second and third column of Table 6.12.

	Detected issues		
	All	**Severe**	**Semantic level**
Registration and first login	3	3	3
Entering personal data, e.g., name, date of birth, etc.	2	0	1
Former studies	4	3	3
Language proficiency	4	3	4
File upload and submission	3	2	1

Table 6.12.: Number of usability issues found using a user-oriented usability evaluation with Thinking Aloud for the applicants portal in the first case study.

Many usability issues, that were detected for the applicants portal by using the user-oriented usability evaluation, were on a semantic level (listed in the fourth column of Table 6.12). This means, that test participants had problems in understanding, e.g., the terminology when entering their grade or did not know how to proceed. Comparing these issues with the smells found using our approach, we did not find any overlap. In addition, one of the issues for the file upload and submission of the application was a statement for the chosen colors of user messages, which also did not correlate to our usability smell detection.

The three remaining issues found through the user-oriented usability evaluation overlapped with our findings. The first is, that two test participants found, that the format of the telephone number entered for the personal data is not validated. Furthermore, several test participants were unsure about the format of the grade of the former studies and made corresponding errors. Finally, 6 test participants had problems with the uploading of files at the end of the process due to the required page reload and scrolling. In addition, one participant found, that the reset button in the forms of the wizard pages seems to have no effect, i.e., it does not provide feedback.

Regarding the reviewer portal, we had informal meetings with some reviewers using it. During these meetings, we checked if some smells detected by our approach correlate with the experience of the reviewers. For example, the reviewers confirmed the missing feedback issue found by our approach, that occurs when switching between different lists of the applications. In these meetings, the comments of the reviewers were also more on a semantic level, which is not addressed by our usability smell detection.

6.3. Case Study 2: Research Website

In the second case study of this thesis, we applied our approach on a website of a research group at the Institute of Computer Science, University of Göttingen [103]. In this section, we first describe the analyzed software and provide several facts about the case study. Then, we list details and results of the task tree generation and the verification of the task tree representativeness applied in this case study. Finally, we show the findings of the usability smell detection and compare them with the results of a user-oriented usability test.

6.3.1. Case Study Facts

The research website analyzed in this case study mainly serves two purposes. First, it provides information about the research done by the research group. This covers group members, research topics, research projects, and publications. Second, the website provides information about study courses offered by the group. This includes organizational course information as well as respective learning material, including slides and exercise sheets.

The website was recorded over a period of almost two years. In between, there was a major structural change of the website, in which no recording took place. Hence, we subdivided the recorded data into data of the *old version* and data of the *new version* of the website, to ensure to not mix the data of the different website structures. For the old version, we recorded more than 3,200 distinct client browsers causing more than 101,000 events in one year and two months. For the new version, we recorded over 1,600 client browsers causing more than 114,000 events in seven months. The details of the number of recorded sessions and events are listed in the upper part of Table 6.13.

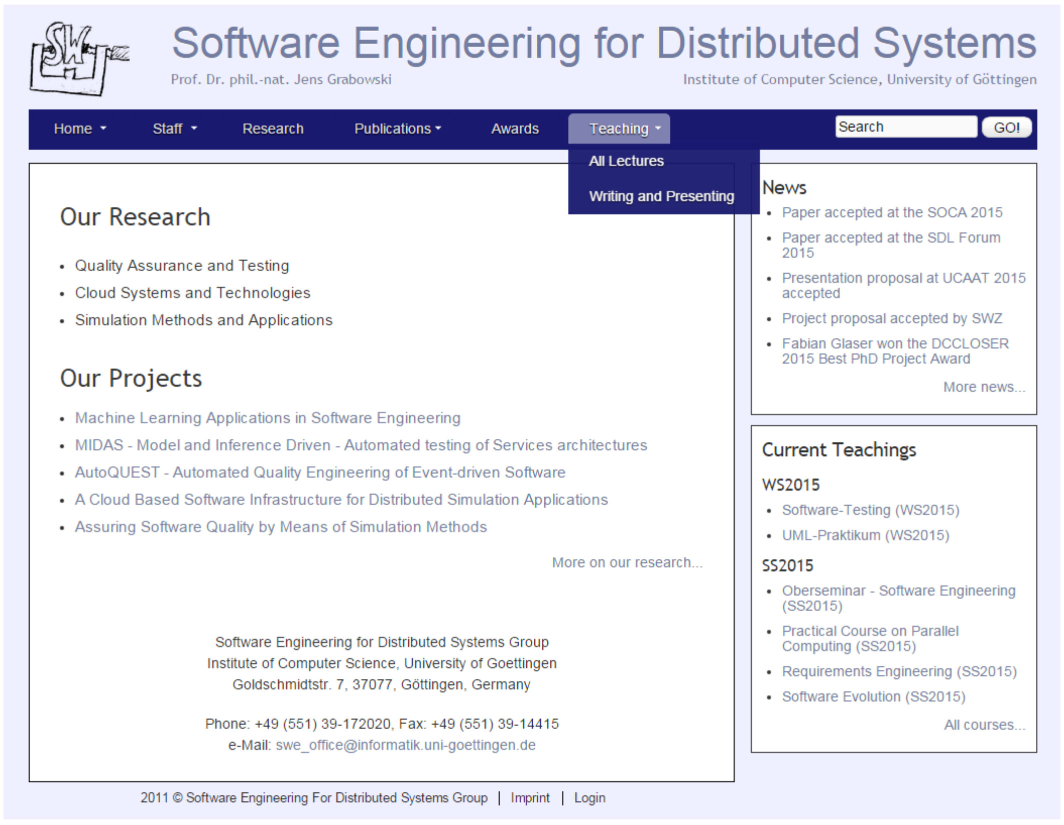
The old and the new version of the website are different, but offer a similar basic structure. A screenshot of the main page of the new version of the website is shown in Figure 6.5. Below the headline, there is the main menu. Some menu items contain sub menus, as shown for the *Teaching* menu entry. On the center stage, there is some initial information

	Old version	**New version**
Recorded data		
Recording period	06/2013 - 08/2014 (15 months)	10/2014 - 04/2015 (7 months)
Events	101,856	114,749
Sessions	13,398	9,627
Distinct clients	3,203	1,665
Post-proc. data		
Events	43,143	43,298
Distinct actions	1,244	864
Sessions	10,620	7,236
Session length μ	4.0	5.9
Session length σ	5.9	8.9

Table 6.13.: Facts of the second case study including recorded and post-processed actions for the old and the new website version.

about the research done by the group, as well as links to the most important projects and contact information. On the right, there are two boxes. The first shows a list of recent news, the second lists the courses currently offered by the research group.

As in the first case study, we parsed the events into AutoQUEST, let AutoQUEST detect common page elements, and post-processed the recorded events to ensure, that each remaining event represents an action instance (see Annex A). Pseudonymization of entered text was not required. Only for login purposes, users need to enter a user name and a password. We did not record both by using the CSS class mechanism offered by AutoQUEST. The resulting number of events, sessions, and distinct actions, that were subsequently used in the task tree generation, are listed for both website versions in the lower part of Table 6.13. Here, we also included the average session length and the standard deviation. In comparison to the first case study, the average session length is much shorter and there are many sessions that contain only one or two events.


Software Engineering for Distributed Systems
Prof. Dr. phil.-nat. Jens Grabowski
Institute of Computer Science, University of Göttingen

Our Research

- Quality Assurance and Testing
- Cloud Systems and Technologies
- Simulation Methods and Applications

Our Projects

- Machine Learning Applications in Software Engineering
- MIDAS - Model and Inference Driven - Automated testing of Services architectures
- AutoQUEST - Automated Quality Engineering of Event-driven Software
- A Cloud Based Software Infrastructure for Distributed Simulation Applications
- Assuring Software Quality by Means of Simulation Methods

More on our research...

Software Engineering for Distributed Systems Group
Institute of Computer Science, University of Goettingen
Goldschmidtstr. 7, 37077, Göttingen, Germany

Phone: +49 (551) 39-172020, Fax: +49 (551) 39-14415
e-Mail: swe_office@informatik.uni-goettingen.de

News

- Paper accepted at the SOCA 2015
- Paper accepted at the SDL Forum 2015
- Presentation proposal at UCAAT 2015 accepted
- Project proposal accepted by SWZ
- Fabian Glaser won the DCCLOSER 2015 Best PhD Project Award

More news...

Current Teachings

WS2015

- Software-Testing (WS2015)
- UML-Praktikum (WS2015)

SS2015

- Oberseminar - Software Engineering (SS2015)
- Practical Course on Parallel Computing (SS2015)
- Requirements Engineering (SS2015)
- Software Evolution (SS2015)

All courses...

2011 © Software Engineering For Distributed Systems Group | Imprint | Login

Figure 6.5.: Screenshot of the homepage of the new version of the research website analyzed in the second case study.

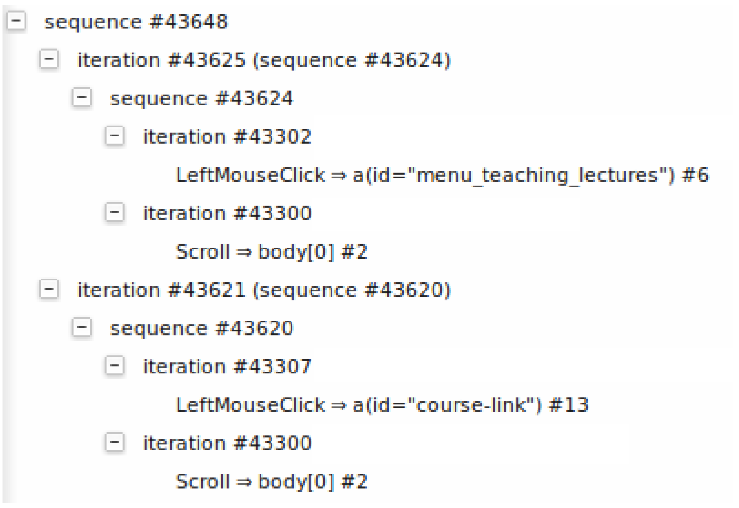
6.3.2. Task Tree Generation Results

As in the first case study, we generated task trees based on the post-processed data using our approach with and without merging of similar sequences. The number of the resulting tasks for the old and the new website version is listed in Table 6.14. The table is structured similar to the corresponding table of the first case study and, hence, not described in more detail. Only a few selections and optionals are created during the merging of task trees in this case study. The number of sequences and iterations before and after the merge are similar. The ratio of covered action instances is similar for both data sets. There are only marginal differences between before and after the merge. When generating task trees for the two data sets in this case study, we did not have task tree generation and merging failures as described in Section 4.4.2.3 and 4.4.3.2.

	Old version	New version
Generated tasks		
Sequences	1,841	1,204
Iterations	566	446
Most prominent sequences	20,5%	20,1%
After merge		
Sequences	1,834	1,191
Iterations	567	445
Selections	1	2
Optionals	3	4
Most prominent sequences	20,2%	20,2%
Action instance coverage		
All sequences	73,8%	73,9%
Most prominent	56,9%	57,8%
Most prominent after merge	56,9%	57,9%

Table 6.14.: Task trees generated in the second case study for the two website versions.

An example for a task tree generated for the homepage of the new website version in Figure 6.5 is shown in Figure 6.6. The represented task shows the typical actions students take to access information about an offered course. First, they click on the menu to open the list of available courses. After some scrolling, they click on the link of a specific course and then scroll down the page.



```
⊟ sequence #43648
  ⊟ iteration #43625 (sequence #43624)
    ⊟ sequence #43624
      ⊟ iteration #43302
          LeftMouseClick ⇒ a(id="menu_teaching_lectures") #6
      ⊟ iteration #43300
          Scroll ⇒ body[0] #2
  ⊟ iteration #43621 (sequence #43620)
    ⊟ sequence #43620
      ⊟ iteration #43307
          LeftMouseClick ⇒ a(id="course-link") #13
      ⊟ iteration #43300
          Scroll ⇒ body[0] #2
```

Figure 6.6.: Example for a task tree generated in the second case study.

6.3.3. Task Tree Representativeness

Identical to the first case study, we evaluated the representativeness of the generated task trees also in the second case study. The details for the created subsets of both data sets can be found in the Table 6.15. This table is subdivided into two subtables, the first containing the subset information for the old, and second for the new website version. The structure of the table is identical to Table 6.4 (see Section 6.2.3).

The number of failures for the task tree generation is seven for the old and four for the new version of the website. There were no merging failures. As there were only few failures,

Old version	**1%**		**2.5%**		**5%**		**10%**		**20%**		**30%**		**40%**		**50%**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Subset statistics																
No. of subsets	50		30		20		10		5		9		6		6	
Generation failures	0		1		3		0		0		1		1		1	
Merging failures	0		0		0		0		0		0		0		0	
Subset contents																
Events	431	0	1,078	0	2,157	0	4,314	1	8,628	1	12,942	0	17,257	0	21,571	0
Distinct actions	118	11	212	11	317	15	461	20	652	13	785	21	895	10	967	20
Generated tasks																
Sequences	28	3	69	6	129	6	244	8	445	9	637	13	827	12	1,014	16
Iterations	23	4	51	6	82	7	135	11	219	12	282	10	342	9	389	12
After merge																
Sequences	28	3	69	6	129	6	244	8	444	9	638	14	825	11	1,012	17
Iterations	23	5	51	6	82	7	135	11	219	11	282	11	343	9	390	12
Selections	0	0	0	0	0	0	0	0	1	1	1	1	1	1	2	1
Optionals	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

New version	**1%**		**2.5%**		**5%**		**10%**		**20%**		**30%**		**40%**		**50%**	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Subset statistics																
No. of subsets	50		30		20		10		5		9		6		6	
Generation failures	1		0		0		0		0		3		0		0	
Merging failures	0		0		0		0		0		0		0		0	
Subset contents																
Events	432	0	1,082	0	2,164	2	4,329	1	8,660	1	12,989	0	17,319	0	21,649	0
Distinct actions	89	12	159	12	232	14	331	16	460	20	541	17	607	12	677	9
Generated tasks																
Sequences	20	3	50	6	94	8	174	6	312	8	432	16	562	21	682	11
Iterations	19	4	38	7	63	10	105	13	165	7	216	12	253	9	298	7
After merge																
Sequences	20	3	50	6	94	8	173	6	311	10	427	15	555	20	673	11
Iterations	19	4	38	7	63	10	105	13	165	7	217	12	253	9	298	7
Selections	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2	1
Optionals	0	0	0	0	0	0	1	1	1	1	2	1	3	1	3	0

Table 6.15.: Information about created subsets, generated task trees, and the comparisons done for the data sets of the second case study.

it was not required to recreate any subsets. The intended average subset size is reached for both data sets with only minimal deviations (standard deviation is at most two action instances). This is because both data sets consist of relatively short sessions.

As in the first case study, the number of detected sequences and iterations is similar for different subsets of the same size. The standard deviation decreases with increasing subset size. This holds true for the task trees before and after the merge. As in the full data sets, the number of detected selections and optionals in the subsets is rather small. At most three optionals and two selections were detected.

In Figure 6.7, we show the plot of the cumulative number of action instances covered by the sequences, which were detected for the new website version before the merge. The plot is created in the same way as the corresponding plot for the first case study and, hence, not described further. As shown in the plot, the most prominent sequences, being 20.1% of all sequences, cover 57.8% of the recorded action instances. All detected sequences cover

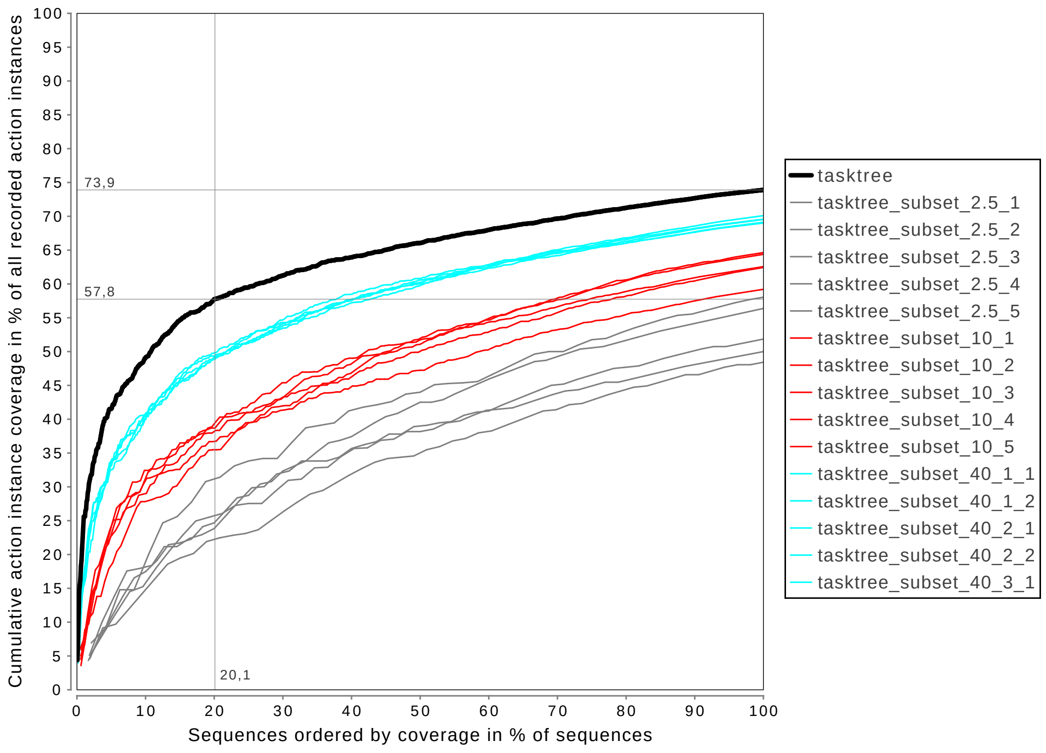


Figure 6.7.: Plot for the cumulative action instance coverage of the unmerged sequences of the new website version (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

73.9% of the recorded action instances. Similar to the first case study, the plot shows, that already a small number of sequences covers a large amount of action instances. However, in this case study, the achieved coverage is not as high as in the first case study. This is because the second case study contains many short sequences, for which no or only a few tasks were detected. Especially for sequences containing only one event, no task trees were detected. The same plot for the data set of the old website version, as well as for the merged task trees for both data sets, are shown in Annex F.1. They demonstrate the same relation between the sequences and the covered action instances, as well as between the subsets and the respective full data set.

Furthermore, we performed the evaluation of the representativeness of the task trees generated for the different subsets. Table 6.16 lists the average ratio (μ) and standard deviation (σ) of sequences transformed into grammars. In comparison to the first case study, the ratio of transformed sequences is higher, which is due to the fact that less, and, hence, also less complex sequences were detected. For both data sets, the ratio of transformed sequences does not decrease with increasing subset size as in the first case study. Also, there is no differences between unmerged and merged sequences. This is because only a few selections and optionals are detected in this case study.

	Old version				**New version**			
	Unmerged		**Merged**		**Unmerged**		**Merged**	
	μ	σ	μ	σ	μ	σ	μ	σ
1%	96.8%	3.8	96.8%	3.8	98.7%	3.0	98.7%	3.0
2.5%	97.3%	2.2	97.3%	2.2	97.9%	2.6	97.9%	2.6
5%	97.9%	1.5	97.9%	1.5	98.6%	1.7	98.6%	1.7
10%	97.7%	1.3	97.7%	1.3	98.5%	1.1	98.5%	1.1
20%	96.9%	0.8	96.9%	0.8	97.4%	0.3	97.3%	0.3
30%	95.5%	1.5	95.5%	1.5	95.8%	0.6	95.7%	0.6
40%	96.0%	0.4	95.9%	0.5	97.1%	1.4	97.1%	1.4
50%	95.5%	0.5	95.4%	0.5	97.3%	0.4	95.4%	1.0

Table 6.16.: Average ratio of sequences that were transformed into grammars for checking their representativeness for other subsets of the same size as well as for the corresponding full data set in the second case study.

After the transformation into grammars, we generated parsers for each subset. As in the first case study, we determined the average number of matches in other subsets of the same size or the respective full data set. We plot these results for the unmerged task trees of the new website version in Figure 6.8. The structure of this plot is already described for the first case study and, therefore, not described in more detail in this chapter. The corresponding plot for the merged task trees, as well as both plots for the old website version can be found in Appendix F.2.

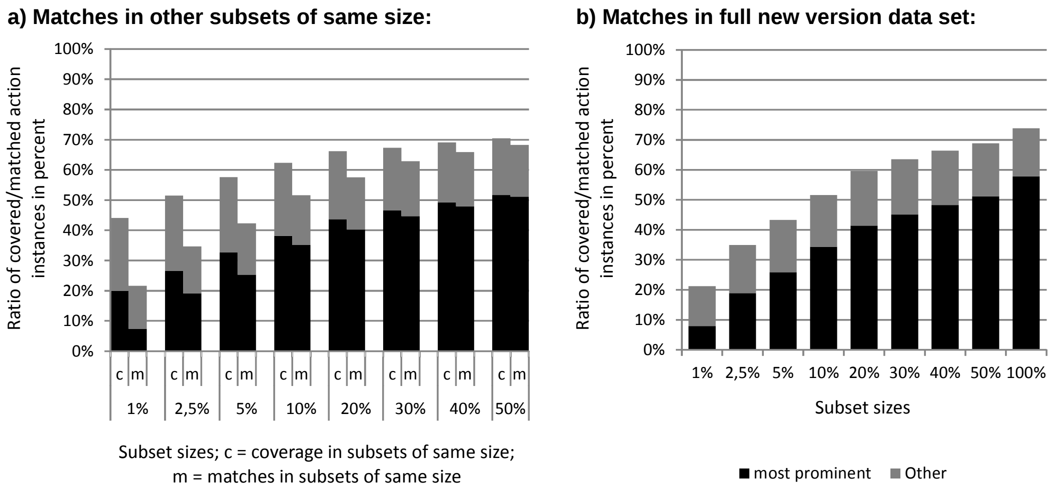


Figure 6.8.: Plot for the action instances matched by parsers, which were generated from unmerged task trees for a specific subset size of the new website version data set in the second case study.

Figure 6.8 shows, that as in the first case study, the average number of matches increases with a higher subset size. This holds true for other subsets of the same size as well as for the full data set. In addition, the ratio of action instances matched solely by the most prominent sequences increases even stronger. In contrast to the first case study, the matches are significantly lower. As with the lower task detection rate, this is caused by the much shorter user sessions in this case study. There is no break-even point after which sequences generated for a subset of a specific size match more action instances in another subset than in the subset from which they were created. This break-even point can again be observed in the evaluation of the old website version (see plot in Appendix F.2).

6.3.4. Usability Evaluation Results

Also in the second case study, we applied our usability smell detection. We performed our analysis on both data sets as well as with unmerged and merged task trees. The resulting numbers of findings for the different usability smells are shown in Table 6.17. The table is structured as for the first case study and, hence, not described further. As in the first case study, we performed a manual inspection of at most 30 smells per smell type, to check if they are true positives or not. In the following subsections, we provide details about the tasks and actions for which smells were found and why we considered the smells as true positives or not.

Old version	**Before merge**					**After merge**				
	Findings		**Inspected**			**Findings**		**Inspected**		
	All	Dupl.	All	Dupl.	True positive	All	Dupl.	All	Dupl.	True positive
Based on task trees										
Important Tasks	219	35%	30	47%	30 (100%)	215	35%	30	47%	30 (100%)
Required Inefficient Actions	1,219	23%	30	27%	30 (100%)	1,215	23%	30	27%	30 (100%)
High GUI Element Distance	1,249	20%	30	47%	7 (23%)	1,242	20%	30	47%	9 (30%)
Missing Feedback	80	-	30	-	2 (7%)	80	-	30	-	2 (7%)
Required Input Method Change	119	31%	30	47%	0 (0%)	114	30%	30	43%	0 (0%)
Missing User Guidance	1	-	1	-	1 (100%)	1	-	1	-	1 (100%)
Based on action instances										
Required Text Format	3	-	3	-	0 (0%)					
Text Input Repetitions	2	-	2	-	0 (0%)					
Text Input Ratio	1	-	1	-	0 (0%)					
Single Checking of Checkboxes	0	-	-	-	-					
Misleading Click Cue	163	-	30	-	7 (23%)					
Required Text Field Focus	1	-	1	-	1 (100%)					
Good Defaults	1	-	1	-	0 (0%)					
Unused GUI Elements	103	-	30	-	0 (0%)					

New version	**Before merge**					**After merge**				
	Findings		**Inspected**			**Findings**		**Inspected**		
	All	Dupl.	All	Dupl.	True positive	All	Dupl.	All	Dupl.	True positive
Based on task trees										
Important Tasks	242	40%	30	50%	30 (100%)	323	28%	30	43%	30 (100%)
Required Inefficient Actions	786	24%	30	30%	30 (100%)	783	24%	30	30%	30 (100%)
High GUI Element Distance	713	23%	30	47%	6 (20%)	700	23%	30	47%	5 (17%)
Missing Feedback	59	-	30	-	2 (7%)	59	-	30	-	2 (7%)
Required Input Method Change	126	35%	30	50%	0 (0%)	118	35%	30	50%	0 (0%)
Missing User Guidance	1	-	1	-	0 (0%)	1	-	1	-	0 (0%)
Based on action instances										
Required Text Format	4	-	4	-	0 (0%)					
Text Input Repetitions	2	-	2	-	0 (0%)					
Text Input Ratio	1	-	1	-	0 (0%)					
Single Checking of Checkboxes	0	-	-	-	-					
Misleading Click Cue	108	-	30	-	7 (23%)					
Required Text Field Focus	1	-	1	-	1 (100%)					
Good Defaults	2	-	2	-	0 (0%)					
Unused GUI Elements	74	-	30	-	0 (0%)					

Table 6.17.: Numbers of detected usability smells in the different data sets of the second case study, including the detection in unmerged and merged task trees.

6.3.4.1. Findings for Usability Smell: Important Tasks

For the usability smell "Important Tasks", we analyzed 30 findings for both data sets. The detailed numbers of findings are listed in Table 6.18. The left column lists the tasks groups. The middle columns list the findings before and after merging task trees for the data set of the old website version. The right columns list the corresponding findings for the new website version.

Findings for usability smell "Important Tasks"	Old version				New version			
	Before merge		After merge		Before merge		After merge	
	Smells	True	Smells	True	Smells	True	Smells	True
Navigation to and presentation of course details	10	10	8	8	8	8	7	7
Navigation to and presentation of staff details	8	8	8	8	7	7	7	7
Navigation to and presentation of publication details	3	3	4	4	4	4	6	6
Navigation to and presentation of research and project details	3	3	3	3	3	3	3	3
Login process	5	5	6	6	8	8	7	7
Combinations of tasks of different groups	1	1	1	1	-	-	-	-

Table 6.18.: Numbers of detected "Important Tasks" usability smells in the two data sets of the second case study.

Also in this case study, we considered all findings for the smell "Important Tasks" as true positives. The reasons for this are the same as in the first case study. The intensities of the inspected findings varied from 0.6 to 4.3. An interesting aspect in this case study is the following. We first evaluated the old version. Then the new version was created and some things were adapted based on our findings for the old version. One change was, that users of the old version could not navigate directly from the homepage to details of a specific course offered by our group. Instead, they had to navigate via an intermediate page. This navigation was referenced by the finding, that had the highest intensity for the old version, i.e., it was executed very often. In the new version, there are direct links to current courses on the homepage of the website. In the evaluation of the new version, a task using these additional links is now referenced by a true positive finding for this smell, what shows, that these links are used rather often and that the corresponding new tasks became important for the users.

6.3.4.2. Findings for Usability Smell: Required Inefficient Actions

For the usability smell "Required Inefficient Actions", we analyzed 30 findings for both data sets. The detailed numbers of the considered smells including the true positives are listed in Table 6.19. We considered all findings for this smell as true positives as the referenced tasks contained at least one inefficient in ten actions. The intensities of the smells were higher in comparison to the first case study. For example, in the data sets of the old version, only 10 findings had an intensity lower than 50%, and only one for the unmerged and two for the merged task trees had less than 34% intensity. For the new version, only 3 findings had an intensity lower than 50%. The lowest intensities were 14.3% for the old, and 25.2% for the new website version.

Findings for usability smell "Required Inefficient Actions"	Old version				New version			
	Before merge		After merge		Before merge		After merge	
	Smells	True	Smells	True	Smells	True	Smells	True
Navigation to and presentation of course details	11	11	11	11	11	11	11	11
Navigation to and presentation of staff details	9	9	9	9	10	10	10	10
Navigation to and presentation of publication details	5	5	4	4	3	3	3	3
Navigation to and presentation of research and project details	3	3	3	3	2	2	2	2
Navigation to the homepage	-	-	-	-	1	1	1	1
Login process	2	2	3	3	3	3	3	3

Table 6.19.: Numbers of detected "Required Inefficient Actions" usability smells in the two data sets of the second case study.

Also in this case study, there were many findings for tasks with only two actions, of which one was inefficient. These were 18 and 19 findings for the unmerged and merged task trees of the old version, as well as 18 for unmerged and merged task trees of the new version. The findings with the highest intensity were for tasks with three actions, of which two were inefficient (6 for the unmerged and merged task trees of the old version, 7 for the unmerged and merged task trees of the new version). In addition, similar to the first case study, we observed, that merged tasks including optional inefficient actions provide more reliable intensities.

One of the major findings based on this smell is, that users often have to scroll when displaying information. Especially, when navigating to a new page, usually the next action is a scroll. This also did not improve in the new website version. The reason may be, that both versions have a rather large header, which cause problems on smaller displays.

6.3.4.3. Findings for Usability Smell: High GUI Element Distance

For the usability smell "High GUI Element Distance", we analyzed 30 findings for both data sets. The detailed numbers of the detected smells for the different task groups are listed in Table 6.20. We considered only 27 of the 120 findings as true positives. Also in this case study, many GUI elements required for a specific task are located close to each other in the corresponding views, i.e., pages of the website. This correlated with the low intensities of the corresponding findings. All but two false positive findings had an intensity of below 0.5.

For the old version, we observed, that for the navigation to details of a specific course, there was a true positive finding for several corresponding tasks. As mentioned, the new website version was adapted to have direct links to courses on the homepage to ease the

Findings for usability smell "High GUI Element Distance"	**Old version**				**New version**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Navigation to and presentation of course details	9	6	9	6	6	3	6	4
Navigation to and presentation of staff details	4	1	4	1	3	0	3	0
Navigation to and presentation of publication details	6	0	6	1	7	2	6	1
Navigation to and presentation of research and project details	-	-	-	-	3	0	4	0
Usage of the search functionalities	1	0	1	0	-	-	-	-
Login process	8	0	7	1	8	0	7	0
Combinations of tasks of different groups	2	0	3	0	3	1	4	0

Table 6.20.: Numbers of detected "High GUI Element Distance" usability smells in the two data sets of the second case study.

access to this information. But still, there were corresponding findings for this smell for the new website version. This is because, the old navigation is still possible and used. For the new tasks utilizing the direct links, there were no findings for this smell.

One issue that we observed in the findings for this smell is, that for the new version there was a task made up of actions caused by a JavaScript and not by the users. For this task, we had findings with a high intensity, which we considered as false positive.

6.3.4.4. Findings for Usability Smell: Missing Feedback

For the usability smell "Missing Feedback", we analyzed 30 findings for both data sets. The detailed numbers of the findings for this smell are listed in Table 6.21. We considered only 8 of the 120 findings as true positives. As for the first case study, the findings were the same for merged and unmerged task trees. The reasons for the rather few true positives are as manifold as in the first case study. We considered many findings as an occasional reuse or

Findings for usability smell "Missing Feedback"	**Old version**				**New version**			
	Before merge		**After merge**		**Before merge**		**After merge**	
	Smells	**True**	**Smells**	**True**	**Smells**	**True**	**Smells**	**True**
Filter for courses listed on website	1	0	1	0	1	0	1	0
Navigation to a specific course	-	-	-	-	2	0	2	0
Download of files provided for a course	1	1	1	1	1	1	1	1
Navigation to and in staff details	3	1	3	1	3	1	3	1
Filter for publications listed on website	1	0	1	0	-	-	-	-
Navigation to news entry	-	-	-	-	1	0	1	0
Other links	24	0	24	0	22	0	22	0

Table 6.21.: Numbers of detected "Missing Feedback" usability smells in the two data sets of the second case study.

a random network latency. This correlated with a rather low intensity of the findings (only 6 for the old and 7 for the new version with an intensity above 35ms). The navigations to the staff details were considered as true positive, as the loading of these pages was partially slow. In the old version, this was caused by a missing caching of these pages on the server side. In the new version, there were some bugs causing the staff pages to become partially large and, hence, slow to be loaded. For both website versions, the finding with the highest intensity (7,715ms for the old, 2,100ms for the new version) was for the download of files provided for a specific course. We considered these smells as true positives, as the download is partially rather slow.

As in the first case study, we observed some findings for different elements of the same page, which are considered semantically equal. For example, there were some findings for links to a specific staff on the staff overview page. These finding could only occur if the users navigated from the staff overview to a concrete staff, then utilized the back button of the browser, and then navigated to the same or other staff. We considered these findings as false positives.

For this smell, the intensities did not provide a separation between true and false positives. The lowest intensity of a true positive for the new version was 1,290ms. In contrast, the highest intensity of a false positive was 1,815ms. Similar values were given for the old version. With only two true positive findings per inspected set, the number of true positives is also rather low and not necessarily representative.

6.3.4.5. Findings for Usability Smell: Required Input Method Change

For the usability smell "Required Input Method Change", we analyzed 30 findings for both data sets. All smells referred either to the login mechanism of the website or the utilization of the search functionality. We considered none of the smells as true positives. The reason is, that both functionalities require the entry of text and can also be utilized with the keyboard only (navigation using the tabulator key is supported). The intensities of the findings varied between 12.5% and 98.4%.

6.3.4.6. Findings for Usability Smell: Missing User Guidance

The intensities of the findings for the smell "Missing User Guidance" in this case study were 3.37 for the old version and 2.35 for the new version of the website. The intensities were the same between merged and unmerged task trees. The intensity for the new version is better than for the old version, which indicates an improvement regarding the user guidance. Unfortunately, the smell requires sessions containing at least 10 action instances to be calculated. But many of the recorded sessions were shorter, causing that in this case study 25,063 of 43,143 processed action instances of the old version and 19,277 of 43,298 processed action instances of the new version were not considered in the calculation of the intensity of this smell. Therefore, the results are not necessarily representative in this case study.

6.3.4.7. Findings for Usability Smell: Required Text Format

For the usability smell "Required Text Format", we analyzed 7 findings in both data sets. They all referred either to the login process or the search functionality of the website. We considered none of the findings as true positives. The reason is, that they referenced text fields, in which no special format is required. This is in line with the very low intensity of the findings of below 8%.

6.3.4.8. Findings for Usability Smell: Text Input Repetitions

For the usability smell "Text Input Repetitions", we analyzed 4 findings and considered none of them as true positives. The reason for two of the findings is, that they were occasional identical entries in the search fields of the website. The other two findings were for the usage of the same user name in the password reset and the login functionality. Here, it could be supported, that the user name, when entered in either of the text fields in a user session, is automatically entered in the other one. But, although being comfortable, due to the sensitivity of user data, this should not be implemented.

6.3.4.9. Findings for Usability Smell: Text Input Ratio

The intensity of the findings for the smell "Text Input Ratio" in this case study was below 1% for both data sets, which is rather low. This is in line with the fact, that text inputs are only required for login and search purposes. Therefore, we considered the findings as false positives.

6.3.4.10. Findings for Usability Smell: Single Checking of Checkboxes

For this smell, there were no findings in this case study. The reason is, that the old and new version of the website did, respectively do not contain any checkboxes. Hence, no findings needed to be analyzed.

6.3.4.11. Findings for Usability Smell: Misleading Click Cue

For this smell, we checked 60 findings, of which we considered 14 (7 for each data set) as true positives. The 60 findings can be subdivided into three groups. The first group covers 30 findings, which showed occasional clicks on the corresponding GUI elements and which we, therefore, considered as false positives. Their intensities were also rather low (below 2%) matching this assessment. The second group, consisting of 16 findings, referred to clicks on headlines. Their intensities reached up to 51.7%. We considered these findings as false positives, although it is partially not clear, why users clicked the headlines. Here, it would have been helpful to have information about how many distinct users clicked the headlines to be able to check if perhaps only a small group of users has such kind of style of using a website. The third group are the 14 true positive findings, whose intensities were between 2.0% and 23.6%. Examples here were clicks on unclickable texts or images. For example, users clicked the image of a staff member on the staff details page, probably, to get a larger version of it, although this functionality is not available. Another example are clicks on names of former staff members, for which no details page exists anymore, and for which, hence, there is no corresponding link.

6.3.4.12. Findings for Usability Smell: Required Text Field Focus

For the smell "Required Text Field Focus", our detection returned one finding per data set, of which both were true positives. The findings were for the login page of the website and showed, that many users placed the cursor into the user name field as first action when opening the page. An inspection of the website showed, that this is actually required as there is no default cursor positioning done on that page. The intensities of these findings were 55.9% and 49.7%.

6.3.4.13. Findings for Usability Smell: Good Defaults

Of the 3 findings for the smell "Good Defaults", we considered none as true positives. This is due to the fact, that, as in the first case study, the findings referred to unrecorded user name and password fields.

6.3.4.14. Findings for Usability Smell: Unused GUI Elements

Also in this case study, the findings for the smell "Unused GUI Elements" were hard to analyze, as they were polluted with GUI elements automatically added by browser plugins. In addition, there were findings for unclicked links in long link lists, such as search results. Furthermore, many findings referenced links inside of longer texts. We did not consider any of the findings as true positives. Only the links in text could perhaps be reduced. But as they do not require additional screen space (the text without the links would cover the same space), these links do not disturb the user's attention.

6.3.5. Result Validation: Application of a User-oriented Usability Test

In this case study, we performed a user-oriented usability evaluation on the old version of the website. This evaluations was similar to the one in the first case study and done in the context of a bachelor thesis [104]. For the new version, no evaluation was done. For the old website version, we started by identifying the typical user groups of the website, which are students, researchers, and other users. Then, we determined the tasks these user groups are likely to perform on the website. This was required, as in contrast to the first case study, there was no single task, like "apply for the master studies", that can be performed by all users. Then we assigned these tasks to the different user groups. Finally, we identified the typical environments, in which the users usually use the website.

After this preparation, we performed a usability evaluation with 28 test participants, of which 20 were students, 5 were researchers, and three were professional web developers, i.e., other users. In each session, which we performed in the typical usage environments of the test participants, we first asked the test participants to fill out a questionnaire for gaining basic information about the test participants for assigning them to a user group. Then, the test participants were asked to perform the identified tasks belonging to the corresponding user group. Finally, the test participants were asked to fill out further questionnaires to get additional information about their opinion on the website.

In sum, each test participant performed seven tasks. Five of them were identical for all user groups. The other two were dependent on the user group of the test participant. For example, a test participant belonging to the user group student had to search for information about a specific course. In contrast, a test participant belonging to the user group researchers had a task of searching for a specific publication. The whole setup, as well as the results, are described in detail in [104] and are, hence, not fully mentioned here again.

The results of the evaluation were, similar to the first case study, mainly on the semantic level. For example, some test participants noted, that a specific term used in the menu was hard to understand or did not match their expectations. Furthermore, six test participants mentioned, that the menu should be located on the left and not on the top. Eight participants did not like the visual design and structuring of the website, e.g., of the start page. Six participants had problems with the fact, that the website is only available in English. Comparing these results with the findings of our smell detection, there is, as in the first case study, not much overlap. The only overlap is, that five test participants experienced the website to be rather slow. This matches our findings for the "Missing Feedback" smell, although the intensity of the findings is rather low.

One major disadvantage of this user-oriented usability evaluation is, that the selected tasks only partially matched the real user tasks. For example, there was no task including a user login on the website. Only one task referred to getting a specific information about a course offered by the research group, but it focused on a course of the previous and not the current semester. Furthermore, four of the selected tasks were not similar or identical to those identified as "Important Task" by our smell detection. An example is the determination of the bus line, that can be used to reach the institute of the research group. This task was analyzed in the user-oriented usability evaluation, but is not one of the major tasks accomplished on the website. Identifying the tasks for a user-oriented usability evaluation more on the basis of the "Important Tasks" smell would perhaps have been more helpful and, probably, would have resulted in more overlap between the results of both evaluations.

6.4. Case Study 3: BORG Calendar App

In our third case study, which is the smallest in this thesis, we analyzed the usage of the Berger-Organizer (BORG) calendar application [105]. In this section, we first describe the analyzed software and provide several facts about the case study. Then, we list details and results of the task tree generation. Finally, we show the findings of the usability smell detection and compare them with the results of a user-oriented usability evaluation.

6.4.1. Case Study Facts

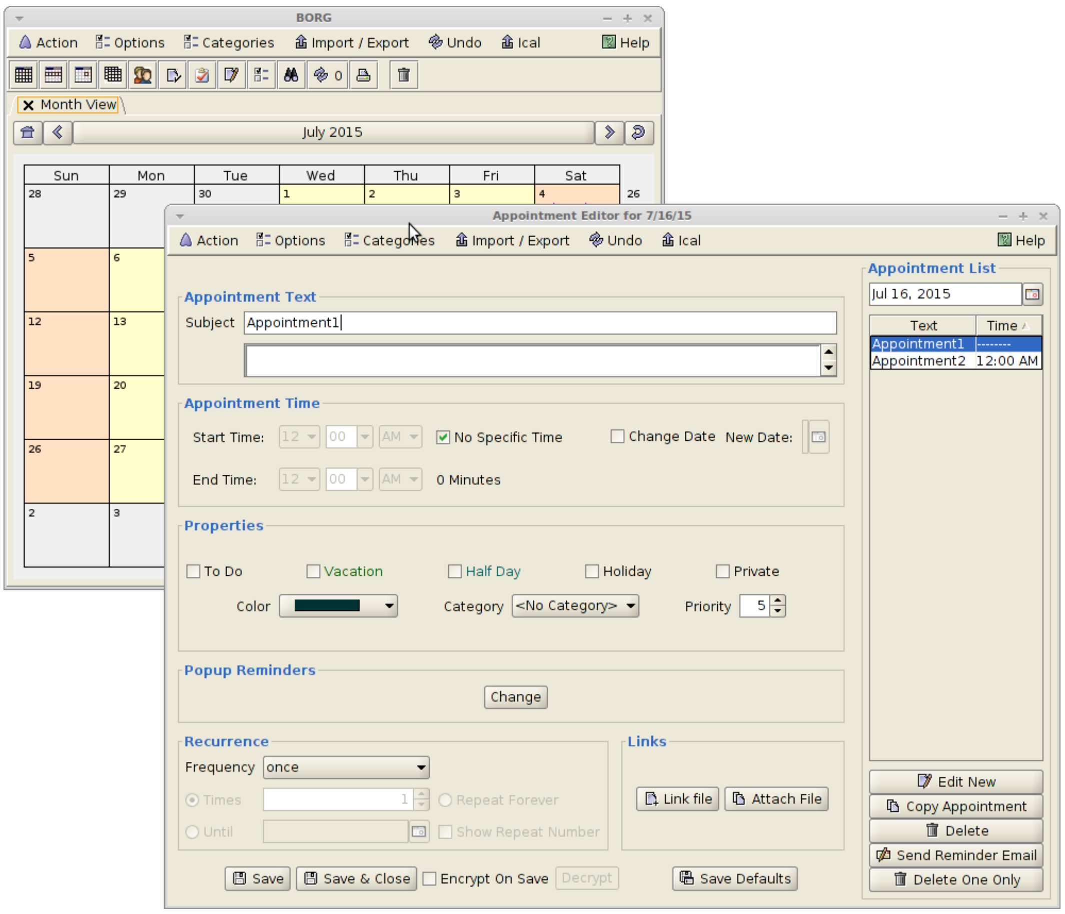
BORG is a powerful tool to organize appointments and tasks. It is written in Java and used as a standalone desktop application. While the overall focus of the tool is more on task management, we solely analyzed the calendar usage. In addition to other functions, the calendar functions of BORG cover:

- entering, changing, and deleting appointments,
- managing and assigning appointment categories,
- appointment repetitions daily, weekly, monthly, and yearly (supports both, always the same day in a month, e.g., always the 10th, or the first of a specific weekday in a month, e.g., the first monday),
- entering of private appointments that can be hidden in the calendar,
- year view, month view, week view, day view, and
- moving of appointments per drag and drop in these views.

A screenshot of BORG is shown in Figure 6.9. It displays the month view (frame in the background) and the appointment editor (frame in the foreground). The month view can be used to get an overview of all appointments in the selected month. The appointment editor is opened, when the user adds a new appointment or changes an existing one. It can be opened as a further tab next to the month view in the main frame or as a standalone frame as shown in the figure. For an appointment, the user can specify a subject, start and end time, several properties including the appointment category, as well as the repetition of the appointment under the point *Recurrence* on the bottom left. The appointment editing is finished using the *Save* button at the bottom. With the *Save & Close* button, the appointment editor is also closed. On the right side of the appointment editor, there is the list of appointments of the selected day. The day can be chosen at the top of the list. The appointments in the list are displayed with subject and start time. If an appointment is a full day appointment, the start time is omitted.

In this case study, we reused recordings of users, which were done in the context of a bachelor thesis [106]. These recordings included log files of 16 users. These users were invited to accomplish five selected goals and to perform corresponding scenarios. These were:

- Create appointments for the birthdays of your parents and add them to an appropriate category.
- Create an appointment on the 5th June 2015 that fits within the already present schedule on that day.
- Handle any appointment collisions in the second week of April 2015.
- Delete the appointment category "Sport".
- Starting with July 2015, create an appointment for a "Group Meeting" that is held monthly.



Subject: Appointment1

Start Time: 12 00 AM ☑ No Specific Time ☐ Change Date New Date:

End Time: 12 00 AM 0 Minutes

☐ To Do ☐ Vacation ☐ Half Day ☐ Holiday ☐ Private

Color: ■ Category: <No Category> Priority: 5

Frequency: once

Times: 1 ☐ Repeat Forever

Until ☐ Show Repeat Number

Jul 16, 2015

Text	Time
Appointment1	--------
Appointment2	12:00 AM

Figure 6.9.: Screenshot of the month view and the view for entering/editing appointments of BORG, which was analyzed in the third case study.

The users were free to decide how to achieve these goals in BORG. This resulted in 16 recorded sessions containing almost 11,500 events. After post-processing the data, 2,537 events remained. The average session length is 158.6 events. The recordings took place in a controlled environment. Therefore, this case study was not an analysis in the field. The details of the recorded data are listed in Table 6.22.

	All sessions
Recorded data	
Recording period	01/2015 - 04/2015 (4 months)
Events	11,448
Sessions	16
Distinct users	16
Post-proc. data	
Events	2,537
Distinct actions	223
Sessions	16
Session length μ	158.6
Session length σ	74.3

Table 6.22.: Facts of the third case study including recorded and post-processed actions for all sessions.

Contrary to the two other case studies, no pseudonymization of the data was required in this case study, as the users did not enter valid personal data. After recording, we parsed the data into AutoQUEST. In contrast to the recording of websites, AutoQUEST is not that elaborated in generating a harmonized GUI model and detecting identical GUI elements in different sessions for Java applications. Instead, AutoQUEST offers a graphical editor for GUI models, in which GUI elements considered distinct after parsing can be marked as identical. We used this editor to manually create a harmonized GUI model. Afterwards, we applied the commands on the parsed events for post-processing as listed in Annex A. Through the post-processing, the number of events decreased as shown in Table 6.22.

6.4.2. Task Tree Generation Results

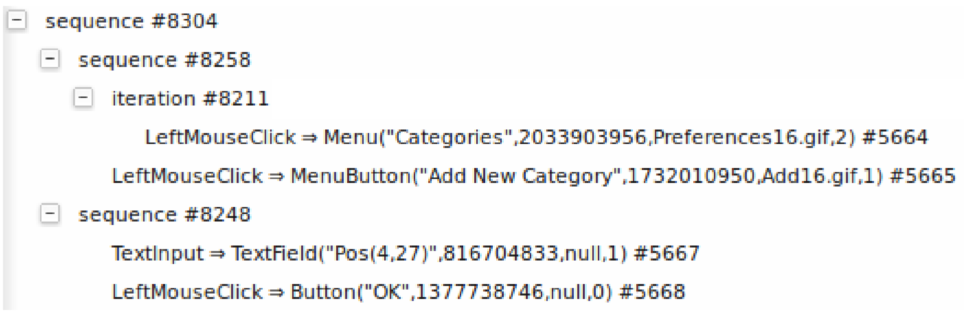
As in the other case studies, we generated task trees based on the post-processed data using our approach with and without merging of similar sequences. As this case study is rather small, no similar sequences were detected during the merge. Hence, the detected tasks before and after the merge were the same. The number of the tasks detected in this case study is listed in the upper part of Table 6.23. The lower part of the table contains the ratio of recorded action instances that are covered by all and by the most prominent sequences.

	Third case study
Generated tasks	
Sequences	194
Iterations	74
Most prominent sequences	22,2%
Action instance coverage	
All sequences	76,4%
Most prominent	47,1%

Table 6.23.: Task trees generated in the third case study.

When generating the task trees in this case study, we did not have task tree generation failures as described in Section 4.4.2.3 and 4.4.3.2.

An example of a task tree generated for the BORG calendar is shown in Figure 6.10. The task tree shows the actions that users take to add a new category to the available appointment categories. First, the users click on the categories menu. Then, they select the menu point to add a category. In a subsequently opening dialog, they enter the category name in a dedicated text field and confirm the dialog using an *OK* button. We did not perform a check for the representativeness of the generated task trees in this case study, as the amount of recorded data is too small.



```
⊟ sequence #8304
    ⊟ sequence #8258
        ⊟ iteration #8211
              LeftMouseClick ⇒ Menu("Categories",2033903956,Preferences16.gif,2) #5664
          LeftMouseClick ⇒ MenuButton("Add New Category",1732010950,Add16.gif,1) #5665
    ⊟ sequence #8248
          TextInput ⇒ TextField("Pos(4,27)",816704833,null,1) #5667
          LeftMouseClick ⇒ Button("OK",1377738746,null,0) #5668
```

Figure 6.10.: Example for a task tree generated for the BORG calendar in the third case study.

6.4.3. Usability Evaluation Results

Also in the third case study, we applied our usability smell detection. As the unmerged and merged task trees in this case study do not differ (see Section 6.4.2), we performed the analysis only on unmerged task trees. The resulting number of findings for the different smell types are shown in Table 6.24. The table is structured as for the previous case studies and, hence, not described further. As in the other case studies, we performed a manual inspection of at most 30 findings per usability smell, to check if they are true positives or not. In the following paragraphs, we provide details about the tasks and actions, for which smells were found, and why we considered the findings as true positives or not.

	Findings		**Inspected**		
	All	Dupl.	All	Dupl.	True positive
Based on task trees					
Important Tasks	194	30%	30	23%	30 (100%)
Required Inefficient Actions	0	-	-	-	-
High GUI Element Distance	160	28%	30	27%	0 (0%)
Missing Feedback	12	-	12	-	2 (17%)
Required Input Method Change	22	9%	22	9%	10 (45%)
Missing User Guidance	1	-	1	-	1 (100%)
Based on action instances					
Required Text Format	1	-	1	-	0 (0%)
Text Input Repetitions	2	-	2	-	0 (0%)
Text Input Ratio	1	-	1	-	0 (0%)
Single Checking of Checkboxes	3	-	3	-	0 (0%)
Misleading Click Cue	0	-	-	-	-
Required Text Field Focus	0	-	-	-	-
Good Defaults	0	-	-	-	-
Unused GUI Elements	9	-	9	-	0 (0%)

Table 6.24.: Numbers of detected usability smells in the third case study.

6.4.3.1. Findings for Usability Smell: Important Tasks

For the usability smell "Important Tasks", we analyzed 30 findings. which we assigned to the task groups as listed in Table 6.25. The table is structured similarly to those of the smell analysis in the second case study. The only difference is, that it is smaller as only one data set and only unmerged task trees are inspected.

For the same reasons as in the previous case studies, we considered all findings for the smell "Important Tasks" as true positives. The intensities of the findings varied from 0.3 to 3.7. We observed, that there were several tasks, in which the users first clicked a GUI

Findings for usability smell "Important Tasks"	**Smells**	**True**
Navigation in the day or month view	17	17
Usage of the appointment editor	9	9
Performing drag and drop of appointments in day or month view	2	2
Exiting the application	1	1
Combination of above tasks	1	1

Table 6.25.: Numbers of detected "Important Tasks" usability smells in the third case study.

element and, afterwards, double-clicked it. This can be a hint, that users already expect a reaction on the first click.

6.4.3.2. Findings for Usability Smell: Required Inefficient Actions

We did not observe findings for the usability smell "Required Inefficient Actions" in this case study. The reason is, that no scrolling was performed by the users. This matches the fact, that all users used the system in a controlled environment on a screen large enough to display the full application GUI. Hence, no scroll bars were shown and used.

6.4.3.3. Findings for Usability Smell: High GUI Element Distance

For the usability smell "High GUI Element Distance", we analyzed 30 findings. The detailed numbers of the detected smells for the different task groups are listed in Table 6.26. We considered none of the 30 findings as true positives. The reason is, that all GUI elements referenced by the findings were rather close to each other. This is also indicated by the intensity of the findings, which was at most 0.5. In addition, for some findings the intensity was relatively high in comparison to the concrete location of the GUI elements. This is caused by the usage of panels in BORG. For example, BORG has parent panels for submenu items. When clicking on a main menu item, e.g., on "Categories" (see Figure 6.9), these panels appear on the screen close to the corresponding main menu item. The panels are the reason, why a submenu item and the corresponding main menu item do not reside in the same direct parent panel. Hence, our GUI element distance metric considers them further away from each other than they actually are.

Findings for usability smell "High GUI Element Distance"	**Smells**	**True**
Navigation in the day or month view	16	0
Usage of the appointment editor	7	0
Change appointment categories	3	0
Exiting the application	1	0
Combination of above tasks	3	0

Table 6.26.: Numbers of detected "High GUI Element Distance" usability smells in the third case study.

6.4.3.4. Findings for Usability Smell: Missing Feedback

For the usability smell "Missing Feedback", we analyzed all 12 findings in this case study. Three of them referenced the navigation in the month view, where two indicated the multiple usage of back and forth buttons to navigate to the next or previous month. As in the first case study, we considered these navigational findings as false positives. The third finding for the month view is for the button, which displays the name of the current month ("July 2015" in Figure 6.9) and which is located between the navigational buttons. This button has no functionality in BORG, but was clicked by some users multiple times. Thereby, it also does not show any reaction. Hence, this was a true positive. Its intensity was 924ms.

The remaining 9 findings were for the usage of the appointment editor. Eight referred to, e.g., the arrows next to combo boxes or to the up and down buttons next to text fields (e.g., the priority text field in Figure 6.9). We did not consider them as true positives, as these buttons are intended to be clicked multiple times. The intensities of these findings varied between 151ms and $1,910$ms. The ninth finding was for the save button on the bottom left. A click on this button stores the appointment. Except the fact, that the entered data disappears in the appointment editor, there is no other visual feedback. Hence, some users clicked the button again. BORG then expects, that a second appointment shall be saved, but denies this due to the fact that the appointment subject is empty. This is not helpful for the user. Hence, we considered this finding as true positive, although its intensity was only 27ms.

6.4.3.5. Findings for Usability Smell: Required Input Method Change

For the usability smell "Required Input Method Change", we analyzed all 22 findings. 18 of them referred to the usage of the appointment editor. Six of the 18 findings referenced tasks, in which the users interacted with the subject text field and, afterwards, with other interaction elements to enter the appointment details. These tasks are not possible to be done with the keyboard alone, as a navigation with the tabulator key from the subject text field to other interaction elements is not fully supported in BORG. Hence, we considered these findings as true positives. Their intensities varied from 30% to 100%. The other findings for the appointment editor referenced tasks that ended with the usage of the subject text field and, hence, did not indicate this issue. Therefore, they were considered as false positives. Their intensities varied between 20% and 100%.

The remaining four findings referred to tasks for adding a new appointment category. This is done by opening a small editor via the "Categories" menu. This editor consists of a text field for the category name, a confirmation, and a cancellation button. When the editor is opened and the user enters a category name, the editor cannot be committed using the enter key. However, this should be supported considering the smell's foundations. Hence, these findings were considered as true positives.

6.4.3.6. Findings for Usability Smell: Missing User Guidance

The intensity of the smell "Missing User Guidance" in the third case study was 4.65. Hence, only about half of the recorded action instances are covered by detected tasks. This may be, because the data set is rather small. Considering, that all participants were asked to achieve the same goals, and that our task detection finds more tasks, the more users perform the same action combinations, then the intensity of this smell should be smaller for an application providing a good user guidance. Hence, we considered this finding as true positive.

6.4.3.7. Findings for Usability Smell: Required Text Format

For the usability smell "Required Text Format", there was only one finding in this case study, which referred to the subject text field in the appointment editor. It was found, because one of the users used one special character when adding an appointment. As there is no required format for this text field, we considered this finding as false positive. This corresponds to the intensity of the finding, which was below 1%.

6.4.3.8. Findings for Usability Smell: Text Input Repetitions

For the usability smell "Text Input Repetitions" there were two findings, both referring to the same two text fields. It is in the nature of this smell, that it always occurs twice. One finding reports, that a text entered in a first text field was also entered in a second text field. The second finding reports, that the same text entered in the second text field was also entered in the first. In this case study, one user occasionally entered the same text twice into two different text fields. Therefore, we considered the findings as false positives. This correlates to the intensities of the findings, which were rather low.

6.4.3.9. Findings for Usability Smell: Text Input Ratio

The intensity of the usability smell "Text Input Ratio" in this case study is rather low with only 3%. This matches the fact that users only entered text for the appointment and category names. Therefore, we considered the finding as false positive.

6.4.3.10. Findings for Usability Smell: Single Checking of Checkboxes

For the smell "Single Checking of Checkboxes", there were three findings in this case study. Two referred to the check boxes in the properties part of the appointment editor: one for the appointment editor as separate frame and one for the appointment editor as additional tab next to the month view. The third finding was for the two check boxes in the appointment time section for the appointment editor as additional tab next to the month view. We considered all findings for this smell as false positives, as the check box groups do not provide mutually exclusive alternatives. The intensities of the findings were also rather low with 3.7%, 3.5%, and 0.3% which matches our decision.

6.4.3.11. Findings for Usability Smell: Misleading Click Cue

In this case study, no user clicked on unclickable text or images. Hence, there was no finding for the smell "Misleading Click Cue". Unfortunately, the smell detection did not react on the unclickable button displaying the current month in the month view. This was due to the fact, that this is a button and no plain text. So from its type, it is a clickable interaction element, but BORG has no implemented functionality for it.

6.4.3.12. Findings for Usability Smell: Required Text Field Focus

There was no finding for the smell "Required Text Field Focus" in this case study. This is correct, as for all views in BORG, that were utilized by the users, the keyboard focus is correctly set to the first text field, when a view is opened. Hence, this is a true negative finding for this smell.

6.4.3.13. Findings for Usability Smell: Good Defaults

Our usability smell detection did not return a finding for the smell "Good Defaults" in this case study. The reason is, that the data set is rather small, and that the individual users utilized different entries for appointment names. In addition, for Java platforms, the selected values of combo boxes are not recorded and, hence, not analyzed by our approach. Finally, the check boxes are only used seldom and, thus, not leading to a statistical significance of their usage.

6.4.3.14. Findings for Usability Smell: Unused GUI Elements

In this case study, the findings for the smell "Unused GUI Elements" referenced many GUI elements of unused functionalities of BORG. We did not consider any of the findings as true positives. The reason for this is, that the data set is rather small, and that the application was used only by a small set of users for a selected set of scenarios. Hence, we could not expect all GUI elements to be used. This smell must be reconsidered after a larger scale usage of the system.

6.4.4. Result Validation: Application of a User-oriented Usability Test

During 10 of the 16 user sessions recorded in this case study, we performed a user-oriented usability evaluation with Thinking Aloud, similar to those in the other case studies. The test participants were asked to perform the above mentioned scenarios. The scenarios were given to them in different orders to prevent learning effects. During the execution of the scenarios, the test participants commented their steps.

Of the 10 test participants, six were male, the others female. The English skills of the test participants were rated by themselves between good to very good. Only one of the test participants saw BORG before, but did not use it. All but one test participants were using other software tools or apps to manage their appointments. The average duration of the test sessions was 20 minutes with a standard deviation of 6 minutes.

The different scenarios are partially overlapping. For example, in several scenarios, a new appointment must be created. Hence, we analyzed the usability issues not by scenario but by all observed data. We identified the following problems the test participants had, while performing the scenarios:

1. Five test participants had problems in finding the way to create an appointment, two needed support by the evaluator.
2. Five test participants had problems in selecting the right monthly repetition for the group meeting appointment, as they did not understand the differences between the available repetition types.
3. Five test participants did not see collisions in appointments, which were actually there.
4. Four test participants had problems, that a newly created appointment category is not instantly visible in the appointment editor if this editor was already opened during the category creation.
5. Four test participants did not understand the function of the save button in the appointment editor.
6. Three test participants did not understand the red indicator attached to some appointments in the right list of the appointment editor.
7. Three test participants where confused by the hiding of private appointments, i.e., they created a new appointment as private and wondered why it is not displayed in the month/day view.
8. One test participant clicked on the central button in the upper part of the month view and wondered why nothing happened.

Considering these results, also in the third case study there are several semantic issues (2, 6, and 7), for which there is no overlap with the findings of our usability smell detection. In addition, issue 3 refers to a functionality that is not directly supported by BORG, as appointment collisions are not detected and marked. Issue 4 is a bug in BORG. Hence, these two findings can also not be in accordance with our usability smell detection.

But there are overlaps with other issues. For example, the first issue is indicated by the findings of the smell "Important Tasks", where users clicked and, afterwards, double clicked the same GUI element. This corresponds to the searching behavior, which we observed in the user-oriented usability evaluation, where users tried to add a new appointment. For this, they clicked a day in the month view and, afterwards, double clicked it, hoping that an appropriate dialog for adding a new appointment shows up, which did not always happen. Issues 5 and 8 match two of the findings for the "Missing Feedback" smell (see above).

6.5. Additional Experiments

Nowadays, software is more and more provided as apps on touch devices as well as in the form of Service Oriented Architectures (SOAs). Hence, our approach should also support the analysis of such setups. AutoQUEST supports the monitoring of software on the Android platform as well as HTTP based SOA applications [100]. In our work, we performed some smaller experiments with these two platforms, which are not worth a full case study.

In a first experiment, we generated task trees for an Android app. For this, we used some example recordings of a small test app, read them into AutoQUEST, and called the task tree generation implementation. A resulting task tree is shown in Figure 6.11. With the monitoring of touch applications, AutoQUEST also introduces a further action type, which is *Touch Single*. This is similar to a click, but indicates, that it was performed as a touch on the screen instead of using a mouse. The example in Figure 6.11 includes two touch actions. First, the users touched on a specific frame. Then they entered a text into a text field. Afterwards, they touched on another frame and entered another text into a further text field. The capabilities of AutoQUEST to analyze Android apps is still in its infancy. Hence, no more in depth case study has been done on this platform and no usability evaluation was performed.

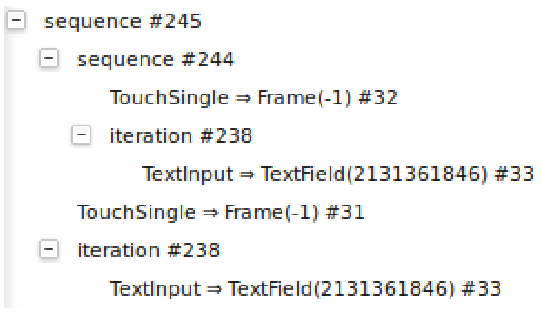

```
sequence #245
  sequence #244
    TouchSingle ⇒ Frame(-1) #32
    iteration #238
      TextInput ⇒ TextField(2131361846) #33
  TouchSingle ⇒ Frame(-1) #31
  iteration #238
    TextInput ⇒ TextField(2131361846) #33
```

Figure 6.11.: Example for a task tree generated for an Android app in the context of additional experiments.

Similarly, AutoQUEST supports the recording of HTTP based SOA applications [107]. Also here, it introduces new event types. In a further experiment, we utilized some test recordings of a SOA, read them into AutoQUEST, and generated corresponding task trees. Also these capabilities of AutoQUEST are not yet well established, which prevented a larger case study.

7. Discussion

In this section, we discuss the results of our case studies and draw conclusions about our approach for task tree generation and usability smell detection. We start by answering the research questions formulated at the beginning of this thesis. Then, we consider strengths and limitations of our approach. Finally, we dedicate a short subsection on ethical aspects to be considered when applying our approach.

7.1. Answers for Research Questions Concerning the Task Tree Generation

In the introduction of this thesis (Section 1.2), we listed several research questions towards the task tree generation which, we answer in this section. The answers are based on the results of our case studies. The first question, RQ 1, asks if typical user tasks can be determined based on recorded action instances and additional information about the structure of the GUI of a software. We addressed this question by applying the task tree generation in several case studies and by checking if generated task trees are representative for the recorded user behavior. By looking at the example task trees generated in the case studies, we consider them as semantically correct and as defining useful action combinations. In addition, the experiments for the representativeness show, that the task trees generated on a subset of recordings are valid descriptions of typical user behavior in other recordings of the same software. Therefore, we answer RQ 1 with yes, it is possible to detect tasks and their corresponding trees by using our methodology. Nevertheless, our case studies also showed, that the task tree generation may not always be possible, as our approach does not always terminate. Yet, this happened rather seldom considering the low number of task tree generation failures of less than 5% observed in all task tree generation attempts performed in the case studies.

The next research question, RQ 1.1, focuses on the level of detail and semantics of the tasks that can be identified. The examples show, that the task trees generated in our case studies represent typical action combinations, which are performed to complete a task. Nonetheless, the corresponding semantic of a task cannot be determined automatically. This still needs to be done by a human. Yet, the task trees provide many details about the actions chosen by users, the typical action combinations, and also execution variants for a task, when considering the merged task trees.

The answer for RQ 1.2, which asks if there are several requirements towards the recorded user actions and the GUI structure to allow for the detection of user tasks, is yes. Our results show, that the task trees become more representative, the more action instances in relation to distinct actions of a software are recorded. This is indicated by the results regarding the coverages of task trees, which were generated for a subset. Furthermore, we see that the recorded events need to represent real action instances. If they instead contain automatically generated events (e.g., the JavaScript generated events in the second case study), the generated task trees do not only represent user actions, but are polluted with technical issues of the monitored software. Regarding the GUI structure, it is important to have a correct GUI model. Otherwise, either tasks are not correctly detected or a subsequent analysis, such as the usability smell detection, may provide incorrect results. It is especially important to be able to identify identical GUI elements also via distinct sessions. In our work, we did this, e.g, by subsequently adding DOM identifiers. Otherwise, action instances are not correctly identified as instances of the same action.

The next research question, RQ 1.3, which asks under which conditions a detected task can still be considered representative for user behavior, cannot be answered with a value of a certain metric. For example, we cannot conclude, that any task covering a specific ratio of recorded action instances is representative. Our case studies show, that more recorded action instances result in more representative task trees. Moreover, we see, that we can subdivide the detected task trees into the most prominent tasks and the other ones, where the most prominent are also the most representative. There is no fixed point regarding the action instance coverage of a task tree, which can be used to distinguish between representative or not. It would be helpful to mark a task on the corresponding coverage plot to see, where it is located, and to decide if it should still be considered representative or not. In addition, one could define a point on the plot where the gradient of the graph falls below a certain threshold, and then consider any sequence lying left of this threshold as representative.

The last research question for the task tree generation, RQ 1.4, is if similar tasks can be detected and merged, and if the merge results are still representative tasks. We can answer both with yes. Especially in the first case study, many similar tasks were detected and merged. In addition, the action instance coverage plots for the merged sequences show, that these are at least as representative as the unmerged task trees. We further found, that fewer recorded action instances and, hence, fewer detected tasks lead to fewer similar tasks that can be merged. For example, in the third case study being the smallest processed data set, no similar tasks were detected at all.

7.2. Answers for Research Questions Concerning the Usability Smell Detection

In the introduction, we formulated several research questions regarding the usability smell detection. The first (RQ 2) was if it is possible to automatically identify usability smells, i.e., indicators for usability issues, in recorded user actions or detected user tasks, considering also additional information about the GUI structure. We addressed this question by applying our usability smell detection in three case studies, by manually inspecting and assessing the findings, as well as by comparing the results with corresponding user-oriented usability evaluations, which are an established method in usability engineering. As we observed true positive usability smell findings as well as several overlaps between these findings and the results of the user-oriented usability evaluation, we answer this question with yes.

The next research question, RQ 2.1, needs to be answered per individual smell. It asks for smell-specific thresholds that should be exceeded, or conditions that should be met, to consider a finding for a usability smell as a true positive. We addressed this question with the same method as the previous question. For the smell "Important Tasks", a threshold can be defined to separate true from false positive findings. The smell aims at detecting the most representative tasks for a data set. Tasks are most representative if they cover a larger number of recorded action instances in comparison to other tasks. Hence, a threshold for this smell is congruent with a threshold for the action instance coverage of the detected tasks. This is in line with the way we calculate the intensity for this smell. The threshold can be defined based on the action instance coverage plots. There are two ways for this. First, a threshold can be a fixed value on the x-axis. All tasks left of this point are considered most representative and, hence, most important. In our work, we used such a fixed point of 20% to separate the most prominent sequences from the other ones. Second, a threshold for this smell can be defined based on the gradient of the graph in an action instance coverage plot. This gradient is initially high and then decreases the less action instances a task covers. A threshold could define a minimum value for this gradient, which is reached by the gradient at some point on the graph. This corresponds to a point on the x-axis, where all tasks left of this point are most important. A definition of this kind of threshold has not been considered in this thesis. Considering the second part of RQ 2.1, there are no conditions that findings of this smell must meet to be considered as true positives.

For the smell "Required Inefficient Actions", we observed a large number of true positive findings. Several findings were compliant with usability issues, which were detected using the established user-oriented usability evaluation. This lets us conclude, that the smell provides valid findings. We derive a threshold for the intensity of this smell to be 10%. This is because the intensities of the true positive findings correlated with our analysis condition, that a task must include at most one inefficient in ten actions. The intensities show, that also values above 50% may be reached for findings of this smell, which lets us conclude, that also higher thresholds may be applicable. But this must be evaluated in further case studies. When solving the usability issues associated with the findings, we propose to address findings with the highest action instance coverage and intensity first. Especially, the results of the first case study show, that for merged task trees findings get a lower threshold if optional inefficient actions are introduced. Hence, findings for this smell are more appropriate, if the smell detection is applied on merged task trees. There were no other conditions for this smell to consider findings as true positives.

In our case studies, many true positive findings for the smell "High GUI Element Distance" had an intensity above 0.5. In contrast, the intensities of almost all false positive findings were below 0.5. The only findings that did not match this limit were two findings for tasks that represented no action instances (second case study) and 24 findings showing the usability issue of required scrolling, which is also indicated by findings for other smells (first case study). Hence, we derive the threshold for the intensity of findings for this smell to be 0.5. This means, that on average, two subsequently required GUI elements of a task are in the same view. There were also some false positives, which were caused by mismatches between the GUI model and the actual rendering of the GUI, as seen in the first and third case study. The intensities of these false positives did not match the actual position of the referenced GUI elements. Considering a threshold of 0.5, this was no issue for the case studies, as the intensities of these findings were below 0.5. Nevertheless, these false positives show, that this smell is sensitive to GUI model issues. Hence, a condition to be met for this smell is a proper GUI model. Due to their compliance with some findings from the user-oriented usability evaluation, we consider findings for this smell as valid indicators for usability issues.

For the smell "Missing Feedback", we considered many findings as false positives. This very often correlated with a low intensity of the findings. However, in the second and the third case study, we also had true positives with a lower intensity than the highest intensity of false positives. Furthermore, the intensity distribution was different for all case studies. For example, in the first case study, the intensities varied between 1ms and $1,032$ms, whereas in the second case study, they varied between 5ms and $7,715$ms. Considering these results, we conclude, that the intensity calculation for this smell provides a good orientation, if a finding is a true positive or not. But based on our data, we cannot conclude on an intensity threshold. This is especially not possible, as the case studies did not provide sufficient true positive findings to have a representative result set. In addition, several findings for this smell in the case studies were false positives due to the usage of common page elements or GUI elements, which are intended to be clicked multiple times. Hence, a corresponding condition must be defined, which must be met to consider the findings of this smell as true positive. In contrast, there were true positive findings, which were compliant with the user-oriented usability evaluations in the case studies. Therefore, we conclude, that findings for this smell, which match the above condition and have a high intensity, provide helpful results. But further investigations are required to improve the detection of this smell and to be able to define a corresponding intensity threshold.

58 findings for the smell "Required Input Method Change" were considered as true positive, which lets us conclude, that this smell in general can be used to identify usability issues. Still, the findings included also many false positives, which were caused by events not representing action instances or by tasks, which represented one variant of text field usage, in which the tabulator key navigation was not used by the users, although possible. Hence, findings for this smell should only be considered as true positives if they refer to a task, for which there is no alternative task, which requires less input method changes. The

intensities of the true and false positive findings varied between similar ranges. Therefore, our case studies did not show a potential threshold for the minimal intensity of the findings. This needs to be determined in future research.

For the smell "Missing User Guidance", there is always one finding per data set. The case studies showed, that the intensities of the findings are a good indicator for the user guidance. For example, in the third case study, the intensity of the finding was relatively high, which matched the experiences from the user-oriented usability evaluation. However, the case studies also showed, that the findings may be biased by the number of recorded action instances and the typical session length as described in the second case study. Anyway, we conclude, that this smell provides valid findings, although an intensity threshold or a scale should be determined in future work.

The results for the smell "Required Text Format" were mostly false positives. The intensities of the findings tended to be higher for true positives than for false positives. In the case studies, there were only few text fields at all, for which a format was required. The true positive findings referenced exactly these text fields. But due to the small number of respective text fields, our case studies did not allow for a representative analysis of findings for this smell. Therefore, we also cannot determine a good threshold for this smell based on our case studies. There were false positive findings for text fields, into which data was entered automatically. This shows, that this smell is sensitive to events not representing real action instances. Hence, we derive a condition for this smell, that it must be applied only for text fields, into which data is entered manually, and that automatic data entries must not be part of the recorded action instances. Furthermore, also text formats well known by users, such as e-mail address formats, are detected, which may lead to false positives. Hence, a further condition for findings of this smell is, that well-known text formats should be ignored. To have an improved validation of this smell, further case studies need to be performed, in which these smell conditions are met, and which include more text fields with required formats.

For the smell "Text Input Repetitions", the findings indicate, that the smell detection can return valid results. Comparing the intensities of the true and false positives, they can be separated from each other by using a threshold of 10%. However, this threshold is only based on the findings in the first case study. The other case studies did not provide sufficient details for a representative analysis. Hence, this threshold must be validated in future research. We did not observe additional conditions that a finding for this smell must meet to consider it as true positive.

The "Text Input Ratio" findings for all case studies correctly indicate the amount of text inputs required in the considered software. Hence, these findings were valid. However, our data does not allow for a conclusion on a threshold or a scale for this smell, as it only contains false, but no true positives. Hence, further case studies are required to determine a concrete threshold for this smell.

The findings for the smell "Single Checking of Checkboxes" in all case studies are not sufficient to draw a clear conclusion about the detection of this smell. In general, the findings referenced the checkbox groups belonging to the case studies and correctly indicated their usage. However, there were no true positives, and all but one finding had low intensities. Hence, further case studies are required to provide more foundation for the assessment of this smell.

For the smell "Misleading Click Cue", there were true positive findings, which lets us conclude, that the detection of the smell provides valid results. Many false positives had a very low action instance coverage or intensity, which signifies, that a certain threshold should be exceeded for both values. The case studies did not provide obvious thresholds for these values. Furthermore, a GUI element should be clicked by several users, so that corresponding findings are representative for a whole user group. But findings for this smell yet do not provide information about how many users performed a click on a GUI element. Therefore, we propose for future work, to determine this number. Then, the findings for this smell can be sorted based on their action instance coverage, their intensity, and the number of users that performed the clicks. Afterwards, the findings can be addressed starting with the findings that have the highest of all values. In further case studies, concrete thresholds should then be identified to consider findings of this smell as true positives. The smell detection also returned false positive findings with a high intensity. These showed, that the smell is sensitive to GUI elements, which are no interaction elements, but made to these through the underlying implementation. In addition, the smell cannot detect interaction elements, that have no underlying functionality, as seen in the third case study. In the second case study, there were findings for headlines, that were clicked several times. Without further information about the number of users that performed these clicks, it was not easy to assess these findings through manual inspection. Hence, this smell should be investigated further in case studies, in which more information is available.

The findings for the smell "Required Text Field Focus" were coherent between the case studies and included many true positives. Therefore, we considered these findings as valid. The intensities of the findings tended to be higher for true positives than for false positives. A concrete threshold for the smell's intensity can be set to 15%, because most false positives had a lower intensity. The only exception here were findings for views, in which the first GUI element is not a text field. This also leads to the following condition to be met by this smell. The findings are only true positives if the logically first interaction element in a view is a text field. In the future, this information can be gained from a the GUI model to automatically filter the findings for this smell.

The smell "Good Defaults" cannot sufficiently be assessed based on our case studies, because only one case study showed true positive findings. Almost all false positives were caused by unrecorded data entries into text fields. Hence, an appropriate filter could reduce the number of false positives significantly. Furthermore, several findings showed good defaults, which should not be used as default in the corresponding software, e.g., due to ethical reasons. Hence, this smell may return findings, which need to be further investigated on a semantic level. This cannot be automated. Overall, the smell requires further investigation in other case studies to ensure, that the findings are valid in other contexts, as well.

The findings for the smell "Unused GUI Elements" were polluted with many GUI elements, which were either not used due to the case study setup (third case study) or due to technical aspects. Therefore, we conclude, that this smell should be further investigated in a more reliable environment to make the findings easier assessable. Nevertheless, the true positives of the first case study can be considered an evidence, that a further investigation is worth the effort.

The next research question regarding the usability smell detection, RQ 2.2, focuses on the conditions, that referenced tasks should match to consider a usability smell as true positive. Based on our findings regarding the task tree representativeness, we conclude, that referenced tasks should be representative, e.g., they should be the most prominent ones. In addition, the tasks must not be based on events that do not represent real action instances. We did not observe significant differences between the findings, which were based on unmerged task trees and those that were based on merged task trees. Only for the smell "Required Inefficient Actions", the number of true and false positives in the first case study was different. Hence, we conclude, that for the usability smell detection both, merged and unmerged task trees can be used. However, in our case studies, we used fixed values for the minimum sequence similarity (75%) and the representativeness of tasks (only the most prominent) to be merged. An adaptation of these values may lead to other merge results and, hence, to other results of the usability smell detection on merged task trees.

The answers to RQ 2.3, which asks for the requirements towards the recorded user actions, the detected tasks, and the information about the GUI structure to allow for an effective usability smell detection, are similar to those for the task detection. Without correctly recorded user actions, events that only represent action instances, and a well-structured GUI model, the findings of our approach can be biased, which leads to potentially wrong conclusions. The preceding paragraphs include detailed answers for this question on a smell-specific level.

The last research question, RQ 2.4, asks if the detection of usability smells is able to replace the application of other usability evaluation methods. Considering the small overlap between the findings of our method and the results from the user-oriented usability evaluation in the three case studies, we can conclude, that our approach provides helpful and valid results also in a large scale, but can only support and not replace the application of other methods. Especially the usability issues on a semantic level, which can be the result of user-oriented usability evaluations, can only partially be detected using our approach.

7.3. Strengths and Limitations

Our approach can be applied on a large scale as shown in the first case study. The analysis of the representativeness of the generated task trees for typical user behavior shows, that based on correct recordings of user actions, we generate representative task trees. Furthermore, the subsequent usability smell detection provides results, which are based on these tasks trees and a large number of recorded action instances and user sessions. Hence, we consider our approach as objective.

When recording actions instances, we intended not to record personal data that was entered into corresponding text fields. Nonetheless, our experiences show, that personal data may also be entered into text fields, which are initially not supposed to be used for personal data. In addition, even a subsequent anonymization and pseudonymization of the data may not be fully correct due to human error, as shown by the remaining personal data in the first case study. Hence, there is a limitation of our approach for correctly encrypting personal data.

Furthermore, the recorded events did not always represent action instances. We performed a post-processing of the recorded data to filter the events for those representing only action instances. Nonetheless, as seen in the usability evaluation, the remaining events still included some, which did not represent action instances.

The detection of task trees is reliable, as we showed, that the task trees represent actual user behavior. An important aspect to be considered is, that we must ensure to record only events that represent action instances. The recording of other events can lead to invalid results of the task tree generation and the usability smell detection. In addition, also the GUI model needs to be complete and correct with respect to the identification of utilized GUI elements. Otherwise, semantically equal actions are considered distinct during the task tree generation and the usability smell detection. This negatively influences the representativeness of the generated task trees and the results of the usability smell detection. Finally, the usability smell detection is reliable only if several smell-specific conditions are met. We mention these conditions together with the corresponding thresholds in the preceding sections. A limitation of the task tree generation is the fact, that it may not terminate and, hence, fail in seldom cases.

Currently, the merging of the task trees relies on the application of Myers diff algorithm. We have not evaluated in our case studies if other diff algorithms yield different merging results. Hence, the results of our case studies that refer to merged task trees may be different if another merging algorithm was applied. In addition, we merge only the most prominent sequences. We have not evaluated, if merging other sequences, as well, would result in differently structured task trees. Finally, our merging is currently done with a threshold for sim_{min} to be 75%. We have not analyzed the potential effects of changing this threshold. Based on these limitations for the task tree merging, we cannot ensure, that with differently merged task trees, there may not be significant differences between the usability smell detection for unmerged and merged task trees.

Our case studies showed, that there may be duplicate findings for a certain usability smell if the smell references tasks. In our manual inspection of findings, we did not evaluate, how many of the true positive findings are duplicates, but mention only the ratio of duplicates in all findings and in the inspected ones. The duplicates may influence our results if they are not equally distributed between true and false positives. For example, if duplicates only occur for true positives, but not for false positives, than the corresponding numbers would not be comparable. In spite of this potential effect, duplicates help to give priorities to usability issues when considering task groups. For example, a higher number of true positive findings for a certain task group than for another is an indicator, that this task group consists of more tasks and is, hence, more important for the user. As such, the number of duplicates serves as a priority for the solving of detected usability issues.

In our case studies, we applied our approach on a large scale for different types of software and different platforms. We validated our results through manual inspection and checked them against the results of established methods. Therefore, we consider our case studies as objective and reliable for the evaluation of our approach. Based on this, we also conclude that our approach is capable of providing valid results. However, our case studies are restricted to websites and a Java application. We cannot draw conclusions for the application of our approach on other platforms or other types of software. In addition, we also recorded events in our case studies that do not represent actual action instances. It was not possible to subsequently filter these events. Hence, we cannot fully be sure if our approach produces other results when working on events that only represent action instances.

An intended benefit of our approach for its user is the reduction of costs for usability evaluation. To apply the approach, its implementation in AutoQUEST can directly be used, as all required infrastructure is already available. Also the effort for the creation of a harmonized GUI model can be minimized if a software already contains correct mechanisms for uniquely identifying GUI elements. Yet, our case studies do not allow to conclude if this cost and effort reduction is really achieved, as we did not perform a cost comparison in this thesis.

Our case studies also showed, that the GUI model can be polluted with invalid GUI elements, e.g., by the environment in which a software is used or by changes in the GUI structure. This is a limitation for the reliability of our results, as they may reference GUI elements not belonging to the analyzed software.

Furthermore, the findings of our approach in our case studies do not necessarily match the results of a user-oriented usability evaluation with thinking aloud. Anyway, they can contribute to findings resulting from the application of other usability evaluation methods. In addition, the data and task trees, that are byproducts of our approach, may support a detailed analysis of usability issues detected with other approaches. Unfortunately, the smells are very generic and, hence, only indicators for possible usability issues. The partially large number of false positives shows, that not any finding for a smell is a valid result. Nonetheless, based on the smell specific conditions and thresholds, that we mention in the previous sections, it should be possible, at least to some degree, to separate true positive from false positive findings automatically.

A further point to pay attention to is, that solutions for findings for a certain usability smell may be counterproductive regarding other aspects of usability. For example, the most efficient task execution may not necessarily be the preferred one, as it may reduce the user's satisfaction or contradict to the user's expectations [95]. Hence, a corresponding finding for a usability smell may indicate a potential for optimizing a user interface in a certain usability aspect. However, this may negatively influence other usability aspects at the same time. In addition, for some findings it may be helpful to have detailed information about groups of users that show a certain behavior. A finding that indicates a usability issue may only apply to a certain user group. Currently, this information can not be derived from the information available for a finding.

7.4. Ethical Aspects

An important aspect of our work is user privacy and security. By using our recording mechanism, we could trace any personal data that is entered by users into forms. This starts with user names and dates of birth, and ends with passwords in login forms. In addition, we would be able to create profiles of individual users concerning their usage behavior on a website. This could include the determination of IP addresses, the utilized browser with all its plugins and settings, as well as typical website navigation. These aspects are another field of investigation and have a big potential for security leaks and for offending the users' privacy.

In our case studies, we analyzed where personal and login data of users were recorded and tried to prevent the recording of this data. In addition to these forecasts, there were further text fields into which partially personal data was entered, although not expected. We encoded this data so that it became unreadable. We sorted the recorded events to sessions and assigned identifiers to them, which do not allow to trace back an individual user. The result were anonymized and pseudomized events grouped to arbitrary sessions.

Based on these events, we compiled models of a general user behavior, but not of individual users. The task tree generation is completely independent of individual user data and profiles. It does also not rely on sessions of a single user and does not consider any user-specific information. Hence, the task trees do not allow to trace back to individual users.

Based on the events and the task trees, we performed the usability smell detection. This also does not require personal or security related data. Only for some smells, like the "Text Input Repetitions" or the "Good Defaults", concrete entered data would result in more helpful findings. This is especially shown for the smell "Required Text Format", where we had several findings based on personal data that remained in the data set even after a careful encoding. But in general, the case studies showed, that also anonymized events and pseudomized text inputs can result in proper findings.

Together, all these considerations show, that our approach does not require to attack user privacy or the security of a website. Anyway, it must be applied with care to not offend the users' privacy. Therefore, it is the responsibility of the person or institution utilizing our approach to apply it with care and to respect the users' privacy. This is an ethical foundation, which applies for any developed software, including those analyzed with our approach.

8. Conclusion

In this section, we conclude the thesis. For this, we provide a short summary and give an outlook on potential future work.

8.1. Summary

In this thesis, we presented an approach for an automated user-oriented usability evaluation in the field. The approach starts with recording instances of user actions on a software as well as determining a corresponding GUI model. Then, it transforms the action instances into task trees to create a model for the usage of the software. Finally, the approach covers a detection of usability smells, which are indicators for usability issues.

For the description of this approach, we first detailed, how a recording of action instances and the derivation of a GUI model is performed. This included also descriptions of the level of detail on which action instances can be and are recorded. Then we described, how to generate task trees from recorded action instances using an n-gram based approach, which was followed by a merging of similar tasks. For this, we also performed a complexity analysis. Afterwards, we depicted, how the approach detects a set of six usability smells based on these task trees and further eight usability smells based solely on the recorded action instances. Together, these smells form a catalog, in which we provided for any smell corresponding foundations and details for their detection.

We implemented our approach and applied it in three case studies on different types of software platforms and interaction concepts. With these case studies, we attempted to answer ten research questions, which we formulated at the beginning of this thesis. These questions focused on the quality and reliability of the task trees created by our approach as well as on the validity of the detected usability smells. The task trees were evaluated by checking their representativeness for different sets of recorded user actions. The findings for the usability smells in the case studies were validated against the results of user-oriented usability evaluations, an established method for usability issue detection. Finally, we discussed our results and drew several conclusions regarding requirements towards recorded action instances, the GUI model of the software, the detected task trees, as well as the usability smells, including conditions for considering findings as true positives.

The findings in the case studies showed, that our approach is able to detect user tasks based on recorded action instances and to generate task trees for them. These task trees are representative for the usage of a software and can be used for further usage analysis. In addition, the case studies showed, that our usability smell detection is capable of providing indicators for usability issues, which are partially compliant with findings of a user-oriented usability evaluation. The case studies had a heterogeneous setup with different types of software and interaction concepts, but still revealed similar results. This allowed us to conclude, that our approach can be applied in a broad variety of contexts.

The approach is fully automated and can, therefore, be applied with minimal effort. In addition, detected usability smells have a reference to the locations of the indicated usability issues and describe potential causes, which allows for their direct solving. As such, the presented approach can easily be applied by persons who do not have a deep understanding of our approach in detail of usability evaluation in general.

8.2. Outlook

Showing promising results, our approach and work open areas for further research. These can be subdivided into the improvement of the task tree generation, the extension of the usability smell detection, further evaluation of our approach, as well as application of our approach in other contexts. This is detailed in the following paragraphs.

The task tree generation currently detects any tasks and does not terminate, although our case studies show, that especially tasks with a low action instance coverage do not seem to be representative for user behavior. In this respect, further research is required to define a stop criterion for the task tree generation, so that only representative tasks remain as result. This includes the development of an approach to estimate a threshold for this criterion. In addition, it would be worthwhile to investigate, how task trees would be different from ours if inefficient actions were considered optional from the beginning and would be ignored in n-grams. Furthermore, our task merging approach currently relies on Myers diff algorithm and a fixed minimal task similarity. Here, further studies could help to check, whether another diff algorithm or another value for the minimal task similarity would create different results. Additionally, we merge only the most prominent sequences, and it should be evaluated if merging further sequences can be beneficial. If these factors influence the merge result, then further studies also need to investigate, if a subsequent usability smell detection on these merged tasks provides other or improved results.

The usability smell detection already includes 14 different types of smells. However, there may be further smells that can be detected, especially if smells for other platforms like touch devices are considered. Hence, in future work the catalog of usability smells should be extended. This extension should also included verified thresholds and detailed conditions, which need to be fulfilled to assess findings as true positives. For this, further smell-specific investigations are required to get improved estimations for thresholds and more refined conditions. In addition, smell-specific extensions are possible. For example, the smell "Required Inefficient Actions" could be extended to detect further inefficient actions, which are not as obvious as scrolling. An example are mandatory actions, like providing a city additionally to a zip code, that users need to perform when filling out a form, but which are unnecessary from the user's perspective. Moreover, if the task tree generation is adapted to detect only representative tasks or to ignore inefficient actions, then the usability smell evaluation needs to be reevaluated to check, whether the same results are achieved. Finally, the smells currently focus on effectiveness and efficiency. There should also be further smells that consider other usability aspects like learnability, error rate of users, or the users' attention.

We expect, that our approach is easy to apply. But in this thesis, we have not investigated if this is really the case. This question can be the focus of further research, as well. In addition, further case studies applying our approach can verify if the approach provides improved results in case its application is planned from the beginning of a software project. For example, we observed, that recorded events must be post-processed. This post-processing may not be required if directly only events are recorded that represent action instances, or if the events are not on the key-stroke level, but on a higher, yet semantic level. Furthermore, our approach must be evaluated in and extended for case studies on other platforms, e.g., with apps on mobile devices or TVs.

A further area of research is the classification of users into groups and whether our task trees allow for that. Such a classification may have a strong impact on the whole evaluation process and the corresponding results. Beyond that, the application of our approach on other events may be of interest. For example, the approach could be used to analyze events caused by developers in source code management systems or events of vocal interfaces. Finally, as being fully automated, our approach can be the basis for an automatic user interface adaptation based on the detected task trees and smells. This would result in highly user and user task optimized systems, where the optimization can focus on users groups or even concrete users. As such, our approach could be the basis for a full personalization of systems for users.

Bibliography

[1] T. Tullis and W. Albert, *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[2] G. Lindgaard and A. Parush, “Utility and experience in the evolution of usability,” in *Maturing Usability*, ser. Human-Computer Interaction Series, E. L.-C. Law, E. T. Hvannberg, and G. Cockton, Eds. Springer London, 2008, pp. 222–240. [Online]. Available: http://dx.doi.org/10.1007/978-1-84628-941-5_10

[3] F. Sarodnick and H. Brau, *Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung*, 1st ed. Huber, Bern, 2006.

[4] K. L. Norman and E. Panizzi, “Levels of automation and user participation in usability testing,” *Interacting with Computers*, vol. 18, no. 2, pp. 246–264, Mar. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.intcom.2005.06.002

[5] F. Paternò, “Tools for remote web usability evaluation,” in *HCI International 2003. Proceedings of the 10th International Conference on Human-Computer Interaction. Vol.1*, vol. 1. Erlbaum, 2003, pp. 828–832, retrieved 7/8/2015. [Online]. Available: http://giove.isti.cnr.it/attachments/publications/2003-A2-95.pdf

[6] M. Y. Ivory and M. A. Hearst, “The state of the art in automating usability evaluation of user interfaces,” *ACM Comput. Surv.*, vol. 33, pp. 470–516, 12 2001. [Online]. Available: http://doi.acm.org/10.1145/503112.503114

[7] A. Dingli and J. Mifsud, “USEFul: A framework to mainstream web site usability through automated evaluation,” *International Journal of Human Computer Interaction (IJHCI)*, vol. 2, no. 1, pp. 10–30, 2011. [Online]. Available: http://cscjournals.org/csc/manuscript/Journals/IJHCI/volume2/Issue1/IJHCI-19.pdf

[8] S. Balbo, S. Goschnick, D. Tong, and C. Paris, “Leading web usability evaluations to WAUTER,” in *The Eleventh Australasian World Wide Web Conference*. Gold Coast, 2005.

[9] A. C. Siochi and R. W. Ehrich, “Computer analysis of user interfaces based on repetition in transcripts of user sessions,” *ACM Trans. Inf. Syst.*, vol. 9, no. 4, pp. 309–335, Oct. 1991. [Online]. Available: http://doi.acm.org/10.1145/119311.119312

[10] D. M. Hilbert and D. F. Redmiles, "Extracting usability information from user interface events," *ACM Comput. Surv.*, vol. 32, no. 4, pp. 384–421, Dec. 2000. [Online]. Available: http://doi.acm.org/10.1145/371578.371593

[11] K. L. Jensen and L. B. Larsen, "Evaluating the usefulness of mobile services based on captured usage data from longitudinal field trials," in *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, ser. Mobility '07. New York, NY, USA: ACM, 2007, pp. 675–682. [Online]. Available: http://doi.acm.org/10.1145/1378063.1378177

[12] F. Paternò, A. Russino, and C. Santoro, "Remote evaluation of mobile applications," in *Task Models and Diagrams for User Interface Design*, ser. Lecture Notes in Computer Science, M. Winckler, H. Johnson, and P. Palanque, Eds. Springer Berlin Heidelberg, 2007, vol. 4849, pp. 155–169. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77222-4_13

[13] S. Rosenbaum, "The future of usability evaluation: Increasing impact on value." in *Maturing Usability*, ser. Human-Computer Interaction Series, E. L.-C. Law, E. T. Hvannberg, and G. Cockton, Eds. Springer, 2008, pp. 344–378. [Online]. Available: http://dblp.uni-trier.de/db/series/hci/LawHC08.html#Rosenbaum08

[14] R. Atterer, "Usability tool support for model-based web development," dissertation, Oktober 2008. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:bvb:19-92963

[15] A. C. Siochi and D. Hix, "A study of computer-supported user interface evaluation using maximal repeating pattern analysis," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '91. New York, NY, USA: ACM, 1991, pp. 301–305. [Online]. Available: http://doi.acm.org/10.1145/108844.108926

[16] G. Cockton, *Usability Evaluation*. Aarhus, Denmark: The Interaction Design Foundation, 2013. [Online]. Available: http://www.interaction-design.org/encyclopedia/usability_evaluation.html

[17] P. Burzacca and F. Paternò, "Remote usability evaluation of mobile web applications," in *Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments*, ser. Lecture Notes in Computer Science, M. Kurosu, Ed. Springer Berlin Heidelberg, 2013, vol. 8004, pp. 241–248. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39232-0_27

[18] H. Trætteberg, *Model-based user interface design*. Information Systems Group, Department of Computer and Information Sciences, Faculty of Information Technol-

ogy, Mathematics and Electrical Engineering, Norwegian University of Science and Technology, May 2002.

[19] P. Harms and J. Grabowski, “Usability of generic software in e-research infrastructures,” *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, vol. 1, no. 3, 2011. [Online]. Available: https://letterpress.uchicago.edu/index.php/jdhcs/article/view/89

[20] P. Harms, S. Herbold, and J. Grabowski, “Trace-based task tree generation,” in *Proceedings of the Seventh International Conference on Advances in Computer-Human Interactions (ACHI 2014)*. XPS - Xpert Publishing Services, 2014.

[21] ——, “Extended trace-based task tree generation,” *International Journal on Advances in Intelligent Systems*, vol. 7, no. 3 and 4, pp. 450–467, 12 2014. [Online]. Available: http://www.iariajournals.org/intelligent_systems/

[22] P. Harms and J. Grabowski, “Usage-based automatic detection of usability smells,” in *Human-Centered Software Engineering*, ser. Lecture Notes in Computer Science, S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, and M. Winckler, Eds. Springer Berlin Heidelberg, 2014, vol. 8742, pp. 217–234. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44811-3_13

[23] ——, “Consistency of task trees generated from website usage traces,” in *Proceedings of the 17th International Conference on System Design Languages (SDL Forum 2015)*. Springer Berlin Heidelberg, 2015.

[24] M. Van Welie, G. C. Van Der Veer, and A. Eliëns, “An ontology for task world models,” in *Proceedings of DSV-IS98, Abingdon*, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.4415

[25] F. Paternò, “ConcurTaskTrees: An engineered notation for task models,” in *The Handbook of Task Analysis for Human Computer Interaction*, D. Diaper and N. Stanton, Eds. Lawrence Erlbaum Associates Publishers, 2003, p. 483–503.

[26] S. Propp, G. Buchholz, and P. Forbrig, “Task model-based usability evaluation for smart environments,” in *Proceedings of the 2Nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams*, ser. HCSE-TAMODIA ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 29–40. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85992-5_3

[27] W. Schweibenz and F. Thissen, *Qualität im Web: Benutzerfreundliche Webseiten durch Usability-Evaluation*, 1st ed. Springer, Berlin, 2003.

[28] “ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (ISO 9241-11:1998),” ISO, 1998.

[29] J. Bosch and N. Juristo, “Designing software architectures for usability,” in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE ’03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 757–758. [Online]. Available: http://dl.acm.org/citation.cfm?id=776816.776937

[30] “ISO 9126-1: Software engineering — Product quality — Part 1: Quality model (ISO 9126-1:2001),” ISO, 2001.

[31] A. Holzinger, “Usability engineering methods for software developers,” *Communications of the ACM*, vol. 48, pp. 71–74, January 2005.

[32] B. Shneiderman, C. Plaisant, M. Cohen, and S. Jacobs, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5th ed. Boston, MA, USA: Pearson Higher Education, 2010.

[33] G. Buscher and R. Biedert, “Usability Testing: Affective Interfaces,” *Informatik-Spektrum*, vol. 33, no. 5, pp. 499–503, oct 2010.

[34] M. Hegner, *Methoden zur Evaluation von Software*, ser. Arbeitsbericht. IZ, InformationsZentrum Sozialwiss., 2003. [Online]. Available: http://books.google.de/books?id=NhnMHAAACAAJ

[35] S. Abrahão, E. Iborra, and J. Vanderdonckt, “Usability evaluation of user interfaces generated with a model-driven architecture tool,” in *Maturing Usability*, ser. Human-Computer Interaction Series, E. L.-C. Law, E. T. Hvannberg, and G. Cockton, Eds. Springer London, 2008, pp. 3–32. [Online]. Available: http://dx.doi.org/10.1007/978-1-84628-941-5_1

[36] L. Kantner, D. H. Sova, and S. Rosenbaum, “Alternative methods for field usability research,” in *Proceedings of the 21st annual international conference on Documentation*, ser. SIGDOC ’03. New York, NY, USA: ACM, 2003, pp. 68–72. [Online]. Available: http://doi.acm.org/10.1145/944868.944883

[37] M. Richter and M. D. Flückiger, *Usability Engineering kompakt: Benutzbare Software gezielt entwickeln*, ser. It Kompakt. Springer Berlin, Heidelberg, 2013.

[38] T. Memmel, “User interface specification for interactive software systems - process-, method- and tool-support for interdisciplinary and collaborative requirements modelling and prototyping-driven user interface specification,” PhD thesis, University of Konstanz, May 2009.

[39] F. Paternò and C. Santoro, “Remote usability evaluation: Discussion of a general framework and experiences from research with a specific tool,” in *Maturing Usability*, ser. Human-Computer Interaction Series, E. L.-C. Law, E. T. Hvannberg,

and G. Cockton, Eds. Springer London, 2008, pp. 197–221. [Online]. Available: http://dx.doi.org/10.1007/978-1-84628-941-5_9

[40] Cátedra SAES de la Universidad de Murcia. (2014) OHT Plus: an application framework for non-intrusive usability testing tools. Retrieved 06/2014. [Online]. Available: http://www.catedrasaes.org/wiki/OHTPlus

[41] K. Renaud and P. Gray, "Making sense of low-level usage data to understand user activities," in *Proceedings of the 2004 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, ser. SAICSIT '04. Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, 2004, pp. 115–124. [Online]. Available: http://dl.acm.org/citation.cfm?id=1035053.1035067

[42] J. H. Kim, D. V. Gunn, E. Schuh, B. Phillips, R. J. Pagulayan, and D. Wixon, "Tracking real-time user experience (true): A comprehensive instrumentation solution for complex systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 443–452. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357126

[43] N. Ramsay, S. Marshall, and A. Potanin, "Annotating UI architecture with actual use," in *Proceedings of the Ninth Conference on Australasian User Interface - Volume 76*, ser. AUIC '08. Darlinghurst, Australia: Australian Computer Society, Inc., 2008, pp. 75–78. [Online]. Available: http://dl.acm.org/citation.cfm?id=1378337.1378351

[44] K. Grooves. (2007) The limitations of server log files for usability analysis. Retrieved 06/2015. [Online]. Available: http://boxesandarrows.com/the-limitations-of-server-log-files-for-usability-analysis/

[45] L. Paganelli and F. Paternò, "Tools for remote usability evaluation of web applications through browser logs and task models," *Behavior Research Methods*, vol. 35, pp. 369–378, 2003.

[46] I. Shah, L. Al Toaimy, and M. Jawed, "RWELS: A remote web event logging system," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 20, pp. 1–11, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1016/S1319-1578(08)80001-8

[47] M. Etgen and J. Cantor, "What does getting WET (web event-logging tool) mean for web usability?" in *5th Conference on Human Factors and the Web - Conference Proceedings*. Gaithersburg, Maryland, USA: NIST, 1999. [Online]. Available: http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html

[48] Drupal.org. (2015) Drupal. Retrieved 06/2015. [Online]. Available: https://www.drupal.org/

[49] G. Buchholz, J. Engel, C. Märtin, and S. Propp, “Model-based usability evaluation - evaluation of tool support,” in *Human-Computer Interaction. Interaction Design and Usability*, ser. Lecture Notes in Computer Science, J. Jacko, Ed. Springer Berlin / Heidelberg, 2007, vol. 4550, pp. 1043–1052.

[50] Piwik.org. (2015) Piwik - liberating analytics. Retrieved 06/2015. [Online]. Available: http://de.piwik.org/

[51] Google. (2015) Google analytics. Retrieved 06/2015. [Online]. Available: http://www.google.com/analytics/

[52] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, M. Florins, and D. Trevisan, “UsiXML: A user interface description language for context-sensitive user interfaces,” in *Proceedings of the ACM AVI'2004 Workshop ”Developing User Interfaces with XML: Advances on User Interface Description Languages”*. Press, 2004, pp. 55–62.

[53] F. Paternò, C. Mancini, and S. Meniconi, “ConcurTaskTrees: A diagrammatic notation for specifying task models,” in *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*, ser. INTERACT '97. London, UK, UK: Chapman & Hall, Ltd., 1997, pp. 362–369.

[54] T. Tiedtke, C. Märtin, and N. Gerth, “AWUSA – a tool for automated website usability analysis,” 2002.

[55] S. Charfi, H. Ezzedine, C. Kolski, and F. Moussa, “Towards an automatic analysis of interaction data for HCI evaluation: Application to a transport network supervision system,” in *Proceedings of the 14th International Conference on Human-Computer Interaction: Design and Development Approaches - Volume Part I*, ser. HCII'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 175–184. [Online]. Available: http://dl.acm.org/citation.cfm?id=2022384.2022407

[56] P. Géczy, N. Izumi, S. Akaho, and K. Hasida, “Usability analysis framework based on behavioral segmentation,” in *E-Commerce and Web Technologies*, ser. Lecture Notes in Computer Science, G. Psaila and R. Wagner, Eds. Springer Berlin Heidelberg, 2007, vol. 4655, pp. 35–45. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74563-1_4

[57] S. Gomez and D. Laidlaw, “Modeling task performance for a crowd of users from interaction histories,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 2465–2468. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208412

[58] S. Amershi, J. Mahmud, J. Nichols, T. Lau, and G. A. Ruiz, “LiveAction: Automating web task model generation,” *ACM Trans. Interact. Intell. Syst.*, vol. 3, no. 3, pp. 14:1–14:23, Oct. 2013. [Online]. Available: http://doi.acm.org/10.1145/2533670.2533672

[59] J.-D. Ruvini and et al., “Ape: Learning user’s habits to automate repetitive tasks,” in *Proceedings of the 2000 Conference on Intelligent User Interfaces*, 2000, pp. 229–232.

[60] A. Cypher, “EAGER: programming repetitive tasks by example,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’91. New York, NY, USA: ACM, 1991, pp. 33–39. [Online]. Available: http://doi.acm.org/10.1145/108844.108850

[61] B. E. John, K. Prevas, D. D. Salvucci, and K. Koedinger, “Predictive human performance modeling made easy,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI ’04. New York, NY, USA: ACM, 2004, pp. 455–462.

[62] S. E. Hudson, B. E. John, K. Knudsen, and M. D. Byrne, “A tool for creating predictive performance models from user interface demonstrations,” in *Proceedings of the 12th annual ACM symposium on User interface software and technology*, ser. UIST ’99. New York, NY, USA: ACM, 1999, pp. 93–102.

[63] A. D’Ulizia, F. Ferri, and P. Grifoni, “A survey of grammatical inference methods for natural language learning,” *Artif. Intell. Rev.*, vol. 36, no. 1, pp. 1–27, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10462-010-9199-1

[64] J. Geertzen and M. Zaanen, “Grammatical inference using suffix trees,” in *Grammatical Inference: Algorithms and Applications*, ser. Lecture Notes in Computer Science, G. Paliouras and Y. Sakakibara, Eds. Springer Berlin Heidelberg, 2004, vol. 3264, pp. 163–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30195-0_15

[65] G. M. Olson, J. D. Herbsleb, and H. H. Rueter, “Characterizing the sequential structure of interactive behaviors through statistical and grammatical techniques,” *Human-Computer Interaction*, vol. 9, pp. 427–472, 1994.

[66] D. Maulsby, “Inductive task modeling for user interface customization,” in *Proceedings of the 2Nd International Conference on Intelligent User Interfaces*, ser. IUI ’97. New York, NY, USA: ACM, 1997, pp. 233–236. [Online]. Available: http://doi.acm.org/10.1145/238218.238331

[67] R. Krimmel, “Improving automatic task tree generation with alignment algorithms,” Göttingen, Germany, 2014.

[68] D. Almeida, J. C. Campos, J. Saraiva, and J. C. Silva, “Towards a catalog of usability smells,” in *ACM SAC 2015 proceedings - Volume I: Artificial Intelligence and Agents, Distributed Systems, and Information Systems*. ACM, 2015, pp. 175–181.

[69] B. E. John. (2015) Cogtool. Retrieved 06/2015. [Online]. Available: http://cogtool.com/

[70] D. D. Salvucci, “Rapid prototyping and evaluation of in-vehicle interfaces,” *ACM Trans. Comput.-Hum. Interact.*, vol. 16, no. 2, pp. 9:1–9:33, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1534903.1534906

[71] S. Feuerstack, M. Blumendorf, M. Kern, M. Kruppa, M. Quade, M. Runge, and S. Albayrak, “Automated usability evaluation during model-based interactive system development,” in *HCSE-TAMODIA ’08: Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 134–141.

[72] M. Quade, M. Blumendorf, and S. Albayrak, “Towards model-based runtime evaluation and adaptation of user interfaces,” in *User Modeling and Adaptation for Daily Routines: Providing Assistance to People with Special and Specific Needs*, 2010.

[73] W. De Pauw and S. Heisig, “Zinsight: A visual and analytic environment for exploring large event traces,” in *Proceedings of the 5th International Symposium on Software Visualization*, ser. SOFTVIS ’10. New York, NY, USA: ACM, 2010, pp. 143–152. [Online]. Available: http://doi.acm.org/10.1145/1879211.1879233

[74] seto GmbH. (2015) m-pathy. Retrieved 06/2015. [Online]. Available: https://www.m-pathy.com/

[75] D. Akers, R. Jeffries, M. Simpson, and T. Winograd, “Backtracking events as indicators of usability problems in creation-oriented applications,” *ACM Trans. Comput.-Hum. Interact.*, vol. 19, no. 2, pp. 16:1–16:40, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2240156.2240164

[76] A. Vargas, H. Weffers, and H. V. da Rocha, “A method for remote and semi-automatic usability evaluation of web-based applications through users behavior analysis,” in *Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research*, ser. MB ’10. New York, NY, USA: ACM, 2010, pp. 19:1–19:5. [Online]. Available: http://doi.acm.org/10.1145/1931344.1931363

[77] T. Carta, F. Paternò, and V. Santana, “Support for remote usability evaluation of web mobile applications,” in *Proceedings of the 29th ACM international conference on Design of communication*, ser. SIGDOC ’11. New York, NY, USA: ACM, 2011, pp. 129–136. [Online]. Available: http://doi.acm.org/10.1145/2038476.2038502

[78] R. Fujioka, R. Tanimoto, Y. Kawai, and H. Okada, “Tool for detecting webpage usability problems from mouse click coordinate logs,” in *Human-Computer Interaction. Interaction Design and Usability*, ser. Lecture Notes in Computer Science, J. Jacko, Ed. Springer Berlin Heidelberg, 2007, vol. 4550, pp. 438–445. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73105-4_48

[79] F. Paternò and G. Ballardin, “RemUSINE: a bridge between empirical and model-based evaluation when evaluators and users are distant.” *Interacting with Computers*, vol. 13, no. 2, pp. 229–251, 2000. [Online]. Available: http://dblp.uni-trier.de/db/journals/iwc/iwc13.html#PaternoB00

[80] F. Paternò, A. Piruzza, and C. Santoro, “Remote usability analysis of multimodal information regarding user behaviour,” 2005, pp. 15–22. [Online]. Available: http://giove.isti.cnr.it/attachments/publications/2005-A2-134.pdf

[81] F. Paternò, A. Piruzza, and C. Santoro, “Remote web usability evaluation exploiting multimodal information on user behavior,” in *Computer-Aided Design of User Interfaces V*, G. Calvary, C. Pribeanu, G. Santucci, and J. Vanderdonckt, Eds. Springer Netherlands, 2007, pp. 287–298. [Online]. Available: http://dx.doi.org/10.1007/978-1-4020-5820-2_24

[82] A. Lecerof and F. Paternò, “Automatic support for usability evaluation,” *IEEE Trans. Softw. Eng.*, vol. 24, pp. 863–888, October 1998.

[83] S. Arondi, P. Baroni, D. Fogli, and P. Mussio, “Supporting co-evolution of users and systems by the recognition of interaction patterns,” in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI ’02. New York, NY, USA: ACM, 2002, pp. 177–186. [Online]. Available: http://doi.acm.org/10.1145/1556262.1556291

[84] J. R. I. Susana Gómez-Carnero, *Evaluation of the Interface Prototypes Using DGAIU Abstract Representation Models*. InTech, 2009-12-01, ch. 29, pp. 517–538.

[85] I. Shah, “Event patterns as indicators of usability problems,” *J. King Saud Univ. Comput. Inf. Sci.*, vol. 20, pp. 31–43, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1016/S1319-1578(08)80003-1

[86] J. Grigera, A. Garrido, and J. Rivero, “A tool for detecting bad usability smells in an automatic way,” in *Web Engineering*, ser. Lecture Notes in Computer Science, S. Casteleyn, G. Rossi, and M. Winckler, Eds. Springer International Publishing, 2014, vol. 8541, pp. 490–493. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08245-5_34

[87] A. Garrido, G. Rossi, and D. Distante, “Refactoring for usability in web applications,” *Software, IEEE*, vol. 28, no. 3, pp. 60–67, May 2011.

[88] *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[89] E. Myers, "An o(nd) difference algorithm and its variations," *Algorithmica*, vol. 1, no. 1-4, pp. 251–266, 1986. [Online]. Available: http://dx.doi.org/10.1007/BF01840446

[90] U. D. of Health and H. Services. (2006) The research-based web design & usability guidelines, enlarged/expanded edition. Retrieved 06/2015. [Online]. Available: http://guidelines.usability.gov/

[91] J. Tidwell, *Designing Interfaces - Patterns for Effective Interaction Design (2. ed.)*, ser. Oreilly Series, M. Treseler, Ed. O'Reilly Media, Incorporated, 2010. [Online]. Available: http://books.google.de/books?id=5gvOU9X0fu0C

[92] P. G. Polson, C. Lewis, J. Rieman, and C. Wharton, "Cognitive walkthroughs: A method for theory-based evaluation of user interfaces," *Int. J. Man-Mach. Stud.*, vol. 36, no. 5, pp. 741–773, May 1992.

[93] X. Ferré, N. Juristo, H. Windl, and L. Constantine, "Usability basics for software developers," *IEEE Softw.*, vol. 18, no. 1, pp. 22–29, Jan. 2001.

[94] D. A. Norman, *The design of everyday things*, 1st ed. [New York]: Basic Books, 2002.

[95] A. Bedford. (2015) Don't prioritize efficiency over expectations. Retrieved 07/08/2015. [Online]. Available: http://www.nngroup.com/articles/efficiency-vs-expectations/

[96] ——. (2015) No more pogo sticking: Protect users from wasted clicks. Retrieved 06/2015. [Online]. Available: http://www.nngroup.com/articles/pogo-sticking/

[97] S. Krug, *Don't make me think!: Web Usability- das intuitive Web*. mitp, 2006.

[98] ——, *Web Usability: Rocket Surgery Made Easy*. Addison Wesley Verlag, 2010.

[99] S. Herbold and P. Harms, "AutoQUEST - Automated Quality Engineering of Event-driven Software," in *Proceedings of the Fourth International Workshop on Testing Techniques & Experimentation Benchmarks for Event-Driven Software*, March 2013, pp. 134 – 139.

[100] U. o. G. Research Group Software Engineering for Distributed Systems, Institute of Computer Science. (2015) AutoQUEST - Testing, Analysing, and Observing Event-driven Software. Retrieved 08/2015. [Online]. Available: http://swe.informatik.uni-goettingen.de/

[101] A. V. Aho, R. Sethi, and J. D. Ullmann, *Compilerbau Teil 1*. Oldenburg Verlag München Wien, 1999.

[102] U. o. G. Institute of Computer Science. (2015) Master application portal - applied computer science. Retrieved 06/2015. [Online]. Available: https://app2.informatik.uni-goettingen.de/cs/2015/wise/accounts/login

[103] U. o. G. Research Group Software Engineering for Distributed Systems, Institute of Computer Science. (2015) Software engineering for distributed systems. Retrieved 08/2015. [Online]. Available: http://swe.informatik.uni-goettingen.de/

[104] F. Trautsch, "User-oriented usability evaluation of a research website," Göttingen, Germany, 2013.

[105] M. Berger. (2015) Borg calendar. Retrieved 07/2015. [Online]. Available: https://mikeberger.github.io/borg_calendar/

[106] D. May, "Evaluation of fully automated usage based testing with mutation testing," Göttingen, Germany, 2015.

[107] S. Herbold, A. D. Francesco, J. Grabowski, P. Harms, L. M. Hillah, F. Kordon, A.-P. Maesano, L. Maesano, C. D. Napoli, F. de Rosa, M. Schneider, N. Tonellotto, M.-F. Wendland, and P.-H. Wuillemin, "The MIDAS Cloud Platform for Testing SOA Applications," in *The 8th IEEE International Conference on Software Testing, Verification and Validation (ICST) 2015 - tools track*, Apr. 2015.

[108] S. Herbold, "Usage-based Testing of Event-driven Software," Ph.D. dissertation, University Göttingen, June 2012 (electronically published on http://webdoc.sub.gwdg.de/diss/2012/herbold/ [retrieved: 1, 2014]), 2012.

[109] H. L. T. H.-C. I. Group. (2015) ConcurTaskTrees Environment. Retrieved 06/2015. [Online]. Available: http://giove.cnuce.cnr.it/ctte.html

Acronyms

AutoQUEST Automated Quality Engineering of Event-driven SofTware.

AWT Abstract Window Toolkit.

AWUSA Automated Website USability Analysis.

BORG Berger-Organizer.

CMS Content Management System.

CRITIQUE Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluations.

CSS Cascading Style Sheets.

DOM Document Object Model.

GOMS Goals, Operators, Methods, and Selection Rules.

GUI Graphical User Interface.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

MFC Microsoft Foundation Classes.

MRP Maximal Repeating Pattern.

ReModEl Remote Model-Based Evaluation.

SOA Service Oriented Architecture.

TaskMODL Task Modelling Language.

UI User Interface.

USEFul	USability Evaluation Framework.
WAUTER	Web Automatic Usability Evaluation EnviRonment.
WebRemUSINE	Web USer INterface Evaluator.
XML	eXtensible Markup Language.

Glossary

GUI element

Any element of a GUI, i.e., all container elements, visual elements, and interaction elements. 9, 10

GUI model

Any element of a GUI including their tree like structure given through the container elements. 9

action

An available interaction of a user with a GUI element, e.g., a click, that can be performed on a button. 10

action instance

The execution of an action. This triggers an event. 10

container element

Element of a GUI that is used for grouping and structuring GUI elements. 9

duplicate

A usability smell finding for a task, whose parent task also has a findings for the same smell. 95

event

Software internal signal, which indicates, that a user performed an action on a GUI element. 10

event target

The GUI element on which the action that caused an event was performed. 10

event type

Type of the action that caused an event, e.g., a mouse click. 10

inefficient action

An action of a user that has no semantic meaning for a task execution, e.g., a horizontal scroll. 10, 53

interaction element

Element of a GUI that is utilized by users, e.g., buttons and text fields. 9, 81

iteration

A type of task, whose temporal relationship defines, that it has only one child, which is executed one or more times. 11

iteration instance

The execution, i.e, the task instance, of an iteration. 12

optional

A type of task, whose temporal relationship defines, that it has only one child, which can be left out. 11

optional instance

The execution, i.e, the task instance, of an optional. 12

selection

A type of task, whose temporal relationship defines, that only one of its children can be executed. 11

selection instance

The execution, i.e, the task instance, of a selection. 12

sequence

A type of task, whose temporal relationship defines, that its children are executed in the given order. 11

sequence instance

The execution, i.e, the task instance, of a sequence. 12

sequence similarity

A similarity metric calculated for the comparison of two sequences. 44

task

A combination of actions performed by users to reach a certain goal [24]. 11

task depth

The number of levels of the task tree that corresponds to a task. 13

task instance

The execution of a task, which forms a tree structure, that is similar to a task tree, with action instances as leaf nodes and task instances as other nodes. 12

task instance list

A list containing task instances and action instances, on which a sequence and iteration detection is done. 36

task list

A list containing actions and tasks. 44

task tree

A tree structure, which is created through the parent-child-relationship defined between tasks and between tasks and actions. 11

temporal relationship

Defines the execution order of the children of a task, which are other tasks or actions. 11

thinking aloud

A method for user-oriented usability evaluation asking users to verbalize their thoughts while performing tasks with a software. 15

usability issue

Problem with the software that decreases the usability. 14

usability smell

Exceptional user behavior that indicates one or more usability issues. 14

view

Container element of a GUI that belongs to a group of container elements, of which only one is visible at a time. 9

visual element

Element of a GUI that displays information to users, but does not allow for interaction. 9

List of Definitions

Variable	Description
A	Set of all actions executable on a software.
a	Individual action $\in A$ executable on a software.
a'	Instance of action $a \in A$ executed on a software.
A'	All action instances executed on a software, which are derived from recorded events.
A^{10}	Multiset containing all executed combinations of 10 subsequently executed actions. The set contains as many duplicates as an action combination was performed by users.
G	All GUI elements belonging to a software.
g	Individual GUI element $\in G$ belonging to a software.
T	Set of all tasks executable on a software.
t	Individual task $\in T$ executable on a software.
t'	Instance of task $t \in T$ executed on a software.
T'	Arbitrary set of task instances executed on a software.
S	Arbitrary set of sequences $S \subseteq T$.
s	Individual sequence.
s'	Instance of Sequence s executed on a software.
S'	Arbitrary set of sequence instances executed on a software.
I	Arbitrary set of iterations $I \subseteq T$.
i	Individual iteration.
i'	Instance of Iteration i executed on a software.
I'	Arbitrary set of iteration instances executed on a software.
O	Arbitrary set of optionals $O \subseteq T$.
o	Individual optional.
o'	Instance of Optional o executed on a software.
O'	Arbitrary set of optional instances executed on a software.
Z	Arbitrary set of selections $Z \subseteq T$.
z	Individual selection.
z'	Instance of Selection z executed on a software.
Z'	Arbitrary set of selection instances executed on a software.
c	Child of a task being another task or an action.
c'	Child of a task instance being a task or action instance.

L	Task list containing actions and tasks.
l	Sublist of a task list.
L'	Task instance list containing action and task instances.
l'	Sublist, i.e., n-gram in a task instance list.
D	List of deltas between two task lists.
d	Individual delta in a delta list D.
i, j	Counter variables.
n, m	Sizes of a list or a set.

Table 8.1.: List of variables used in formulas in this thesis.

Function	Description
$x(a)$	Returns all instances of an action a.
$viewActionInstances(A', view)$	Returns sublists of subsequent action instances $a' \in A'$ that were executed in the *view*.
$c(t)$	Ordered list of children $c_1 \dots c_n$ of a task t, where $c_i \in A \cup T \setminus \{t\}$.
$\|c(t)\|$	Number of children of a task t.
$c(t')$	Ordered list of children $c'_1 \dots c'_n$ of an instance t' of task t, which are other task or action instances.
$\|c(t')\|$	Number of children of an instance t' of task t.
$depth(t)$	Returns the depth of a task t.
$x(t)$	Returns all instances of a task t.
$L(t)$	Returns the task list belonging to a task t.
$L'(t')$	Returns the task instance list belonging to a task instance t'.
$p(l')$	Returns the position of an n-gram l' in a task instance list.
$a(t)$	Returns the average number of actions performed for a task t, when t is executed with a minimum number of action instances.
$a(L)$	Returns the average number of actions performed for the tasks $t \in L$ plus the number of actions $a \in L$.
$a(d)$	For a delta d, returns the average number of actions performed for the tasks covered by d plus the number of actions covered by d.
$a'(t)$	Returns all action instances based on which a task t was generated.
$a'(t')$	Returns the action instances representing the instance t' of task t.
$ia'(t')$	Number of instances of inefficient actions performed in the task instance t'.
$\|d\|$	Number of actions belonging to a delta between two task lists.
$sim(s_1, s_2)$	Metric for the similarity of two sequences s_1 and s_2.
$G(s')$	Returns the GUI elements used in the instances of efficient actions, which were performed in the sequence instance s'.
$dist(g_1, g_2)$	Distance between two GUI elements g_1 and g_2 based on a GUI model.
$R_{smell}(t)$	Intensity of the usability smell *smell* calculated for task t.
$R_{smell}(t')$	Intensity of the usability smell *smell* calculated for task instance t'.

Table 8.2.: List of functions used in formulas in this thesis.

List of Figures

List of Algorithms

List of Listings

List of Tables

A. AutoQUEST Commands for Post-Processing Recorded Events

Recorded events can have characteristics specific for a certain platform or case study. For example, there may be a difference between the level of detail of recorded events (see Section 4.2), or events can have a wrong order. In this annex, we describe commands for AutoQUEST, which we implemented to post-process the data recorded in our case studies. We further list, which of the commands were applied in which case study.

A.1. Extended AutoQUEST Commands

The first command, which we implemented, focuses on the detection of mouse clicks, mouse drag and drops, and mouse double clicks. The command is named *condense mouse clicks*. Depending on the recorded platform, a single mouse click is either recorded as

- one event representing the mouse click,
- two events, the first representing the mouse button down and the second the mouse button up, or
- three events, the first representing the mouse button down, the second the mouse button up, and the third the mouse click itself.

The corresponding command handles these different situations and replaces several events if present by a single event representing a mouse click. In addition, the command considers drag and drops, which are also represented by two events, the first being a mouse button down and the second being a mouse button up. If the command finds two such subsequent events, and if these events are recorded on different GUI elements, they are replaced by a single mouse drag and drop event instead of a mouse click event. Finally, the command ensures, that two subsequent mouse click events, even if detected in the steps before, are replaced by a single mouse double click event. This is only done if the difference between the events' timestamps is smaller than 500 milliseconds. The 500 milliseconds are a typical value when considering attempts for a double click on Windows machines.

The event recordings of Java Swing/AWT applications as well as of websites include many focus change events. These indicate the GUI element that has the keyboard focus at a specific point in time. For example, there is usually a focus change event to a specific text field before any text is entered into this field. These events are implicit to the user actions, but they do not represent a real user action. Instead, they are logged in combination with another event, e.g., a mouse click on a text field, that implies the focus change. To reduce the amount of processed events, we implemented an AutoQUEST command that searches for the focus change events and adapts the target of subsequent keyboard events to be the same as the preceding focus change event. Afterwards, the focus change events are dropped. The command is named *correct key interaction targets*.

Especially on websites, the event recordings have a wrong order of events with respect to the usage of the tabulator for navigating between GUI elements of a form. For example, when a user enters a text into a text field and then uses the tabulator key for finishing the entry and navigating to the next text field, the recorded events are first the pressing of the tabulator key and second the change of the text in the first text field. But from a logical point of view, the entering of the text precedes the pressing of the tabulator key. Therefore, we implemented a command named *correct tab key navigation order* in AutoQUEST, which switches two subsequent events representing this wrong order.

Furthermore, when recording users on websites, a user may have opened several browser windows or tabs for the same website. When working in both windows at the same time, the recorded events are send as groups, together with the identical client id, to the monitoring server. Due to the intermediate client side caching and its separation in the different browser windows, the monitoring server will not receive the events in the order in which they were actually performed. But due to the same client id, they are stored in the same log file. In addition, the events may represent action combinations, which would not be possible if the website was opened only once. Therefore, we implemented a command in AutoQUEST that searches for sessions in which the events are not ordered according to their timestamps. If this command, which we call *check event time stamps*, finds such sessions, it drops them so that they are ignored in our subsequent analysis. The command does not fully prevent, that there may still be a wrong event order caused by multiple parallel browser tabs. But due to the large amounts of recorded data, we expect that these wrong orders become noise. Furthermore, the events that were logged together are in the correct order.

Events recorded on Java Swing/AWT applications are on key stroke level. Hence, when recording text inputs, the AutoQUEST library records individual key presses on the keyboard, which leads to a large amount of events for entering text. Therefore, we implemented two commands to combine these events into single events for each text input. The first command is called *sort key interactions*. This checks the individual usage of keys and sorts them in a harmonized way. For example, a recording may include the following events for entering the text "OK" into a text field:

- press 'O' key
- press 'K' key
- release 'O' key
- release 'K' key

This order is correct but less expected than if the second and the third event were switched. The command *sort key interactions* detects the above wrong order and switches the second and third event. Similarly, it creates a harmonized event order, when special keys like shift are used to type upper case characters or other signs. The second command, which we called *detect text input*, afterwards processes the sorted key stroke level events. It determines those event groups that together represent the entering of a text into a text field. These events are then replaced by a single text input event. In addition, the command determines the entered text based on the pressed keys. This text becomes an additional parameter of the text input event.

A.2. AutoQUEST Command Application

The above commands are applied differently in the distinct case studies. This is because the case studies utilize different platforms and, hence, require different commands. In all case studies, we applied the commands *correct key interaction targets*, *correct tab key navigation order*, and *condense mouse clicks*. In the first case study, it was not required to execute other commands for post-processing the events, as we did not observe the issues solved with these commands in the data. In addition, on websites we already recorded full text inputs. Hence, no detection of the entered text based on keystrokes is required.

In the second case study, we additionally applied the command *check event time stamps*. This was required, as we observed the issue of badly sorted events in our data due to the opening of multiple tabs by the same client at the same time. In the third case study, we additionally applied the commands *sort key interactions* and *detect text input*. This was necessary, as on Java platforms, AutoQUEST does not record full text entries but the single key strokes. The application of other commands was not required.

B. Extension of GUI Models for Websites With DOM Ids

Websites often lack ids for the different DOM elements. But these ids are helpful when creating a GUI model, as they can help to identify the same GUI elements in different sessions and log files. If a website does not make use of such ids, AutoQUEST allows to subsequently add these ids again, when log files of a website are parsed into AutoQUEST. For this, the parsing process can be equipped with a parameter file, which contains a mapping between a GUI element of a website to an id. In this file, a GUI element of a website is identified by its path through the DOM of a page. To have this path as unambiguous as possible, it contains ids of other GUI elements of a page as well as indexes of the elements in in the child lists of their parents. An example for such a path with a mapping to the id *image1* is the following:

```
document(path\=/path1)/html[0]/body[0]/table(htmlId\=tab_1)/tr[2]/td[3]/img[0] = image1
```

This path identifies a GUI element on the page of a website which is available at the path */path1*. The element itself belongs to the table with the HTML id *tab_1* and is located in the third row and fourth column. The element itself is the first image located in the indicated table cell. Using this mapping, the derivation of a GUI model as described in Section 4.3 can be done. The configuration files that we applied in the first and the second case study, which were both websites, are listed in the following subsections.

B.1. Parsing Configuration for Case Study 1

```
#######################################################################################
# unify document paths
#######################################################################################

document(path\=/cs14/accounts/login/)=CLEAR_QUERY,accounts/login

document(path\=/cs2013ws/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2013ws/accounts/password/reset/)=accounts/password_reset
document(path\=/cs2013ws/review/$)=CLEAR_QUERY,review/overview
document(path\=/cs2013ws/review/#/$)=review/details
document(path\=/cs2013ws/review/##/$)=review/details
document(path\=/cs2013ws/review/###/$)=review/details
document(path\=/cs2013ws/review/####/$)=review/details
document(path\=/cs2013ws/review/#/commit_history)=review/details_history
document(path\=/cs2013ws/review/##/commit_history)=review/details_history
document(path\=/cs2013ws/review/###/commit_history)=review/details_history
document(path\=/cs2013ws/review/####/commit_history)=review/details_history
document(path\=/cs2013ws/review/#/view)=review/details_view
document(path\=/cs2013ws/review/admin/)=review/admin
document(path\=/cs2013ws/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/cs2013ws/review/distribute/)=review/distribute
document(path\=/cs2013ws/review/export/)=review/export
document(path\=/cs2013ws/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/cs2013ws/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2013ws/review/statistics)=review/statistics

document(path\=/cs2014ss/accounts$)=accounts
document(path\=/cs2014ss/accounts/activate/########################################/$)=accounts/activate
document(path\=/cs2014ss/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2014ss/accounts/new)=accounts/new
document(path\=/cs2014ss/accounts/password/change/)=accounts/password_change
document(path\=/cs2014ss/accounts/password/reset/$)=accounts/password_reset
document(path\=/cs2014ss/accounts/password/reset/confirm/###########################/$)=accounts/password_reset_confirm
document(path\=/cs2014ss/accounts/register/$)=accounts/register
document(path\=/cs2014ss/accounts/register/complete/)=accounts/register_complete
document(path\=/cs2014ss/adm)=adm
document(path\=/cs2014ss/application/$)=application
document(path\=/cs2014ss/application/edit/contact)=application/edit_contact
document(path\=/cs2014ss/application/edit/degree/$)=application/edit_degree
document(path\=/cs2014ss/application/edit/degree/additional-degree/$)=application/edit_degree_additional-degree
document(path\=/cs2014ss/application/edit/degree/additional-degree/#/)=application/edit_degree_additional-degree_details
document(path\=/cs2014ss/application/edit/degree/additional-degree/##/)=application/edit_degree_additional-degree_details
document(path\=/cs2014ss/application/edit/degree/additional-degree/###/)=application/edit_degree_additional-degree_details
document(path\=/cs2014ss/application/edit/degree/additional-university/#/)=application/edit_degree_additional-university_details
document(path\=/cs2014ss/application/edit/degree/additional-university/##/)=application/edit_degree_additional-university_details
document(path\=/cs2014ss/application/edit/degree/additional-university/###/)=application/edit_degree_additional-university_details
document(path\=/cs2014ss/application/edit/degree/degree/)=application/edit_degree
document(path\=/cs2014ss/application/edit/degree/eu/)=application/edit_degree_eu
document(path\=/cs2014ss/application/edit/degree/university/)=application/edit_degree_university
document(path\=/cs2014ss/application/edit/english/$)=application/edit_english
document(path\=/cs2014ss/application/edit/english/english/)=application/edit_english
document(path\=/cs2014ss/application/edit/english/en-other/)=application/edit_english_en-other
document(path\=/cs2014ss/application/edit/english/en-test/)=application/edit_english_en-test
document(path\=/cs2014ss/application/edit/german/$)=application/edit_german
document(path\=/cs2014ss/application/edit/german/de-nat/)=application/edit_german_de-nat
document(path\=/cs2014ss/application/edit/german/de-other/)=application/edit_german_de-other
document(path\=/cs2014ss/application/edit/german/de-test/)=application/edit_german_de-test
document(path\=/cs2014ss/application/edit/german/german/)=application/edit_german
document(path\=/cs2014ss/application/edit/gsv/$)=application/edit_gsv
document(path\=/cs2014ss/application/edit/gsv/gsv/)=application/edit_gsv
document(path\=/cs2014ss/application/edit/gsv/gsv-document/)=application/edit_gsv_gsv-document
document(path\=/cs2014ss/application/edit/personal)=application/edit_personal
document(path\=/cs2014ss/application/file/)=application/file
document(path\=/cs2014ss/application/withdraw/)=application/withdraw
document(path\=/cs2014ss/auth/user/)=auth/user
document(path\=/cs2014ss/imprint/)=imprint
document(path\=/cs2014ss/review/$)=CLEAR_QUERY,review/overview
document(path\=/cs2014ss/review/#/$)=review/details
document(path\=/cs2014ss/review/##/$)=review/details
document(path\=/cs2014ss/review/###/$)=review/details
document(path\=/cs2014ss/review/####/$)=review/details
document(path\=/cs2014ss/review/#/view$)=review/details_view
document(path\=/cs2014ss/review/##/view$)=review/details_view
document(path\=/cs2014ss/review/###/view$)=review/details_view
document(path\=/cs2014ss/review/####/view$)=review/details_view
document(path\=/cs2014ss/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ss/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ss/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ss/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ss/review/admin/$)=review/admin
document(path\=/cs2014ss/review/admin/#$)=review/admin_details
document(path\=/cs2014ss/review/admin/##$)=review/admin_details
document(path\=/cs2014ss/review/admin/add)=review/admin_add
document(path\=/cs2014ss/review/admin/auth$)=review/admin_auth
document(path\=/cs2014ss/review/admin/auth/user)=review/admin_auth_user
document(path\=/cs2014ss/review/admin/deadline)=review/admin_deadline
document(path\=/cs2014ss/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/cs2014ss/review/distribute/)=review/distribute
document(path\=/cs2014ss/review/export/)=review/export
document(path\=/cs2014ss/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/cs2014ss/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2014ss/review/statistics)=review/statistics
document(path\=/cs2014ss/wizard/$)=wizard
document(path\=/cs2014ss/wizard/additional-degree/$)=wizard/additional-degree
document(path\=/cs2014ss/wizard/additional-degree/#/$)=wizard/additional-degree_details
document(path\=/cs2014ss/wizard/additional-degree/##/$)=wizard/additional-degree_details
document(path\=/cs2014ss/wizard/additional-university/#/$)=wizard/additional-university_details
document(path\=/cs2014ss/wizard/additional-university/##/$)=wizard/additional-university_details
document(path\=/cs2014ss/wizard/degree/)=wizard/degree
document(path\=/cs2014ss/wizard/de-nat/)=wizard/de-nat
document(path\=/cs2014ss/wizard/de-other/)=wizard/de-other
document(path\=/cs2014ss/wizard/de-test/)=wizard/de-test
document(path\=/cs2014ss/wizard/english/)=wizard/english
document(path\=/cs2014ss/wizard/en-nat/)=wizard/en-nat
document(path\=/cs2014ss/wizard/en-other/)=wizard/en-other
document(path\=/cs2014ss/wizard/en-test/)=wizard/en-test
document(path\=/cs2014ss/wizard/eu/)=wizard/eu
document(path\=/cs2014ss/wizard/german/)=wizard/german
document(path\=/cs2014ss/wizard/gsv/)=wizard/gsv
document(path\=/cs2014ss/wizard/gsv-document/)=wizard/gsv-document
document(path\=/cs2014ss/wizard/personal/)=wizard/personal
document(path\=/cs2014ss/wizard/university/)=wizard/university

document(path\=/cs2014ws/accounts$)=accounts
document(path\=/cs2014ws/accounts/activate/########################################/$)=accounts/activate
document(path\=/cs2014ws/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2014ws/accounts/password/change/)=accounts/password_change
document(path\=/cs2014ws/accounts/password/reset/$)=accounts/password_reset
document(path\=/cs2014ws/accounts/password/reset/confirm/##########################/$)=accounts/password_reset_confirm
document(path\=/cs2014ws/accounts/password/reset/confirm/###########################/$)=accounts/password_reset_confirm
document(path\=/cs2014ws/accounts/register/$)=accounts/register
document(path\=/cs2014ws/accounts/register/complete/)=accounts/register_complete
document(path\=/cs2014ws/adm)=adm
document(path\=/cs2014ws/application/$)=application
document(path\=/cs2014ws/application/edit/contact)=application/edit_contact
document(path\=/cs2014ws/application/edit/degree/$)=application/edit_degree
document(path\=/cs2014ws/application/edit/degree/additional-degree/$)=application/edit_degree_additional-degree
document(path\=/cs2014ws/application/edit/degree/additional-degree/#/)=application/edit_degree_additional-degree_details
document(path\=/cs2014ws/application/edit/degree/additional-degree/##/)=application/edit_degree_additional-degree_details
document(path\=/cs2014ws/application/edit/degree/additional-degree/###/)=application/edit_degree_additional-degree_details
document(path\=/cs2014ws/application/edit/degree/additional-university/#/)=application/edit_degree_additional-university_details
document(path\=/cs2014ws/application/edit/degree/additional-university/##/)=application/edit_degree_additional-university_details
document(path\=/cs2014ws/application/edit/degree/additional-university/###/)=application/edit_degree_additional-university_details
document(path\=/cs2014ws/application/edit/degree/degree/)=application/edit_degree
document(path\=/cs2014ws/application/edit/degree/eu/)=application/edit_degree_eu
document(path\=/cs2014ws/application/edit/degree/university/)=application/edit_degree_university
document(path\=/cs2014ws/application/edit/english/$)=application/edit_english
document(path\=/cs2014ws/application/edit/english/english/)=application/edit_english
document(path\=/cs2014ws/application/edit/english/en-nat/)=application/edit_english_en-nat
document(path\=/cs2014ws/application/edit/english/en-other/)=application/edit_english_en-other
document(path\=/cs2014ws/application/edit/english/en-test/)=application/edit_english_en-test
document(path\=/cs2014ws/application/edit/german/$)=application/edit_german
document(path\=/cs2014ws/application/edit/german/de-nat/)=application/edit_german_de-nat
document(path\=/cs2014ws/application/edit/gsv/$)=application/edit_gsv
document(path\=/cs2014ws/application/edit/gsv/gsv-document/)=application/edit_gsv_gsv-document
document(path\=/cs2014ws/application/edit/personal)=application/edit_personal
document(path\=/cs2014ws/application/edit/st/$)=application/edit_st
document(path\=/cs2014ws/application/edit/st/special-treatment/$)=application/edit_st_special-treatment
document(path\=/cs2014ws/application/edit/st/special-treatment-document/$)=application/edit_st_special-treatment-document
document(path\=/cs2014ws/application/withdraw/)=application/withdraw
document(path\=/cs2014ws/auth/user/)=auth/user
document(path\=/cs2014ws/imprint/)=imprint
document(path\=/cs2014ws/cs2014ws/review/$)=CLEAR_QUERY,review/overview
document(path\=/cs2014ws/review/$)=CLEAR_QUERY,review/overview
document(path\=/cs2014ws/review/#/$)=review/details
document(path\=/cs2014ws/review/##/$)=review/details
document(path\=/cs2014ws/review/###/$)=review/details
document(path\=/cs2014ws/review/####/$)=review/details
document(path\=/cs2014ws/review/#/view$)=review/details_view
document(path\=/cs2014ws/review/##/view$)=review/details_view
document(path\=/cs2014ws/review/###/view$)=review/details_view
document(path\=/cs2014ws/review/####/view$)=review/details_view
document(path\=/cs2014ws/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ws/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ws/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ws/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014ws/review/#/commit_history$)=review/details_history
document(path\=/cs2014ws/review/##/commit_history$)=review/details_history
document(path\=/cs2014ws/review/###/commit_history$)=review/details_history
document(path\=/cs2014ws/review/####/commit_history$)=review/details_history
document(path\=/cs2014ws/review/#/interview$)=review/details_interview
document(path\=/cs2014ws/review/##/interview$)=review/details_interview
document(path\=/cs2014ws/review/###/interview$)=review/details_interview
document(path\=/cs2014ws/review/####/interview$)=review/details_interview
document(path\=/cs2014ws/review/admin/$)=review/admin
document(path\=/cs2014ws/review/sdmin$)=invalid-page
document(path\=/cs2014ws/review/admin/#$)=review/admin_details
document(path\=/cs2014ws/review/admin/##$)=review/admin_details
document(path\=/cs2014ws/review/admin/###$)=review/admin_details
document(path\=/cs2014ws/review/admin/add)=review/admin_add
document(path\=/cs2014ws/review/admin/deadline)=review/admin_deadline
document(path\=/cs2014ws/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/cs2014ws/review/distribute/)=review/distribute
document(path\=/cs2014ws/review/export/)=review/export
document(path\=/cs2014ws/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/cs2014ws/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2014ws/review/statistics)=review/statistics
document(path\=/cs2014ws/wizard/$)=wizard
document(path\=/cs2014ws/wizard/additional-degree/$)=wizard/additional-degree
document(path\=/cs2014ws/wizard/additional-degree/#/$)=wizard/additional-degree_details
document(path\=/cs2014ws/wizard/additional-degree/##/$)=wizard/additional-degree_details
document(path\=/cs2014ws/wizard/additional-university/#/$)=wizard/additional-university_details
document(path\=/cs2014ws/wizard/additional-university/##/$)=wizard/additional-university_details
document(path\=/cs2014ws/wizard/bologna/)=wizard/bologna
document(path\=/cs2014ws/wizard/degree/)=wizard/degree
document(path\=/cs2014ws/wizard/de-nat/)=wizard/de-nat
document(path\=/cs2014ws/wizard/de-other/)=wizard/de-other
document(path\=/cs2014ws/wizard/de-test/)=wizard/de-test
document(path\=/cs2014ws/wizard/english/)=wizard/english
document(path\=/cs2014ws/wizard/en-nat/)=wizard/en-nat
document(path\=/cs2014ws/wizard/en-other/)=wizard/en-other
document(path\=/cs2014ws/wizard/en-test/)=wizard/en-test
document(path\=/cs2014ws/wizard/eu/)=wizard/eu
document(path\=/cs2014ws/wizard/german/)=wizard/german
document(path\=/cs2014ws/wizard/gsv/)=wizard/gsv
document(path\=/cs2014ws/wizard/gsv-document/)=wizard/gsv-document
document(path\=/cs2014ws/wizard/special-treatment/)=wizard/special-treatment
document(path\=/cs2014ws/wizard/special-treatment-document/)=wizard/special-treatment-document
document(path\=/cs2014ws/wizard/personal/)=wizard/personal
document(path\=/cs2014ws/wizard/university/)=wizard/university

document(path\=/cs2014wsnb/accounts$)=accounts
document(path\=/cs2014wsnb/accounts/$)=accounts
document(path\=/cs2014wsnb/accounts/activate/########################################/$)=accounts/activate
document(path\=/cs2014wsnb/accounts/login/$)=CLEAR_QUERY,accounts/login
document(path\=/cs2014wsnb/accounts/login/CometBirdHTML%5CShell%5COpen%5CCommand)=invalid-page
document(path\=/cs2014wsnb/accounts/password/change/)=accounts/password_change
document(path\=/cs2014wsnb/accounts/password/reset/$)=accounts/password_reset
document(path\=/cs2014wsnb/accounts/password/reset/confirm/##########################/$)=accounts/password_reset_confirm
document(path\=/cs2014wsnb/accounts/password/reset/confirm/###########################/$)=accounts/password_reset_confirm
document(path\=/cs2014wsnb/accounts/register/$)=accounts/register
document(path\=/cs2014wsnb/accounts/register/complete/)=accounts/register_complete
document(path\=/cs2014wsnb/adm)=adm
document(path\=/cs2014wsnb/application/$)=application
document(path\=/cs2014wsnb/application/edit/contact)=application/edit_contact
document(path\=/cs2014wsnb/application/edit/degree/$)=application/edit_degree
document(path\=/cs2014wsnb/application/edit/degree/additional-degree/$)=application/edit_degree_additional-degree
document(path\=/cs2014wsnb/application/edit/degree/additional-degree/#/)=application/edit_degree_additional-degree_details
document(path\=/cs2014wsnb/application/edit/degree/additional-degree/##/)=application/edit_degree_additional-degree_details
document(path\=/cs2014wsnb/application/edit/degree/additional-degree/###/)=application/edit_degree_additional-degree_details
document(path\=/cs2014wsnb/application/edit/degree/additional-university/#/)=application/edit_degree_additional-university_details
document(path\=/cs2014wsnb/application/edit/degree/additional-university/##/)=application/edit_degree_additional-university_details
document(path\=/cs2014wsnb/application/edit/degree/additional-university/###/)=application/edit_degree_additional-university_details
document(path\=/cs2014wsnb/application/edit/degree/degree/)=application/edit_degree
document(path\=/cs2014wsnb/application/edit/degree/eu/)=application/edit_degree_eu
document(path\=/cs2014wsnb/application/edit/degree/university/)=application/edit_degree_university
document(path\=/cs2014wsnb/application/edit/english/$)=application/edit_english
document(path\=/cs2014wsnb/application/edit/english/english/)=application/edit_english
document(path\=/cs2014wsnb/application/edit/english/en-nat/)=application/edit_english_en-nat
document(path\=/cs2014wsnb/application/edit/english/en-other/)=application/edit_english_en-other
document(path\=/cs2014wsnb/application/edit/english/en-test/)=application/edit_english_en-test
document(path\=/cs2014wsnb/application/edit/german/$)=application/edit_german
document(path\=/cs2014wsnb/application/edit/german/de-nat/)=application/edit_german_de-nat
document(path\=/cs2014wsnb/application/edit/german/de-other/)=application/edit_german_de-other
document(path\=/cs2014wsnb/application/edit/german/de-test/)=application/edit_german_de-test
document(path\=/cs2014wsnb/application/edit/german/german/)=application/edit_german
document(path\=/cs2014wsnb/application/edit/gsv/$)=application/edit_gsv
document(path\=/cs2014wsnb/application/edit/gsv/gsv-document/)=application/edit_gsv_gsv-document
document(path\=/cs2014wsnb/application/edit/personal)=application/edit_personal
document(path\=/cs2014wsnb/application/edit/st/$)=application/edit_st
document(path\=/cs2014wsnb/application/edit/st/special-treatment/$)=application/edit_st_special-treatment
document(path\=/cs2014wsnb/application/edit/st/special-treatment-document/$)=application/edit_st_special-treatment-document
document(path\=/cs2014wsnb/application/file/)=application/file
document(path\=/cs2014wsnb/application/withdraw/)=application/withdraw
document(path\=/cs2014wsnb/auth/user/)=auth/user
document(path\=/cs2014wsnb/cs2014wsnb/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2014wsnb/cs2014wsnb/review/)=CLEAR_QUERY,review/overview
document(path\=/cs2014wsnb/imprint/)=imprint
document(path\=/cs2014wsnb/review/$)=CLEAR_QUERY,review/overview
document(path\=/cs2014wsnb/review/#/$)=review/details
document(path\=/cs2014wsnb/review/##/$)=review/details
document(path\=/cs2014wsnb/review/###/$)=review/details
document(path\=/cs2014wsnb/review/####/$)=review/details
document(path\=/cs2014wsnb/review/#/view$)=review/details_view
document(path\=/cs2014wsnb/review/##/view$)=review/details_view
document(path\=/cs2014wsnb/review/###/view$)=review/details_view
document(path\=/cs2014wsnb/review/####/view$)=review/details_view
document(path\=/cs2014wsnb/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014wsnb/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014wsnb/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014wsnb/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs2014wsnb/review/#/commit_history$)=review/details_history
document(path\=/cs2014wsnb/review/##/commit_history$)=review/details_history
document(path\=/cs2014wsnb/review/###/commit_history$)=review/details_history
document(path\=/cs2014wsnb/review/####/commit_history$)=review/details_history
document(path\=/cs2014wsnb/review/#/interview$)=review/details_interview
document(path\=/cs2014wsnb/review/##/interview$)=review/details_interview
document(path\=/cs2014wsnb/review/###/interview$)=review/details_interview
document(path\=/cs2014wsnb/review/####/interview$)=review/details_interview
document(path\=/cs2014wsnb/review/admin/$)=review/admin
document(path\=/cs2014wsnb/review/sdmin/$)=invalid-page
document(path\=/cs2014wsnb/review/admin/#$)=review/admin_details
document(path\=/cs2014wsnb/review/admin/##$)=review/admin_details
document(path\=/cs2014wsnb/review/admin/###$)=review/admin_details
document(path\=/cs2014wsnb/review/admin/add)=review/admin_add
document(path\=/cs2014wsnb/review/admin/deadline)=review/admin_deadline
document(path\=/cs2014wsnb/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/cs2014wsnb/review/distribute/)=review/distribute
document(path\=/cs2014wsnb/review/export/)=review/export
document(path\=/cs2014wsnb/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/cs2014wsnb/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs2014wsnb/review/statistics)=review/statistics
document(path\=/cs2014wsnb/wizard/$)=wizard
document(path\=/cs2014wsnb/wizard/additional-degree/$)=wizard/additional-degree
document(path\=/cs2014wsnb/wizard/additional-degree/#/$)=wizard/additional-degree_details
document(path\=/cs2014wsnb/wizard/additional-degree/##/$)=wizard/additional-degree_details
document(path\=/cs2014wsnb/wizard/additional-university/#/$)=wizard/additional-university_details
document(path\=/cs2014wsnb/wizard/additional-university/##/$)=wizard/additional-university_details
document(path\=/cs2014wsnb/wizard/bologna/)=wizard/bologna
document(path\=/cs2014wsnb/wizard/degree/)=wizard/degree
document(path\=/cs2014wsnb/wizard/de-nat/)=wizard/de-nat
document(path\=/cs2014wsnb/wizard/de-other/)=wizard/de-other
document(path\=/cs2014wsnb/wizard/de-test/)=wizard/de-test
document(path\=/cs2014wsnb/wizard/english/)=wizard/english
document(path\=/cs2014wsnb/wizard/en-nat/)=wizard/en-nat
document(path\=/cs2014wsnb/wizard/en-other/)=wizard/en-other
document(path\=/cs2014wsnb/wizard/en-test/)=wizard/en-test
document(path\=/cs2014wsnb/wizard/eu/)=wizard/eu
document(path\=/cs2014wsnb/wizard/german/)=wizard/german
document(path\=/cs2014wsnb/wizard/gsv/)=wizard/gsv
document(path\=/cs2014wsnb/wizard/gsv-document/)=wizard/gsv-document
document(path\=/cs2014wsnb/wizard/special-treatment/)=wizard/special-treatment
document(path\=/cs2014wsnb/wizard/special-treatment-document/)=wizard/special-treatment-document
document(path\=/cs2014wsnb/wizard/personal/)=wizard/personal
document(path\=/cs2014wsnb/wizard/university/)=wizard/university

document(path\=/cs/2015/suse/accounts$)=accounts
document(path\=/cs/2015/suse/accounts/activate/########################################/$)=accounts/activate
document(path\=/cs/2015/suse/accounts/login/$)=CLEAR_QUERY,accounts/login
document(path\=/cs/2015/suse/accounts/password/change/)=accounts/password_change
document(path\=/cs/2015/suse/accounts/password/reset/$)=accounts/password_reset
document(path\=/cs/2015/suse/accounts/password/reset/confirm/##########################/$)=accounts/password_reset_confirm
document(path\=/cs/2015/suse/accounts/password/reset/confirm/###########################/$)=accounts/password_reset_confirm
document(path\=/cs/2015/suse/accounts/register/$)=accounts/register
document(path\=/cs/2015/suse/accounts/register/complete/)=accounts/register_complete
document(path\=/cs/2015/suse/adm)=adm
document(path\=/cs/2015/suse/application/$)=application
document(path\=/cs/2015/suse/application/edit/contact)=application/edit_contact
document(path\=/cs/2015/suse/application/edit/degree/$)=application/edit_degree
document(path\=/cs/2015/suse/application/edit/degree/additional-degree/$)=application/edit_degree_additional-degree
document(path\=/cs/2015/suse/application/edit/degree/additional-degree/#/)=application/edit_degree_additional-degree_details
document(path\=/cs/2015/suse/application/edit/degree/additional-degree/##/)=application/edit_degree_additional-degree_details
document(path\=/cs/2015/suse/application/edit/degree/additional-degree/###/)=application/edit_degree_additional-degree_details
document(path\=/cs/2015/suse/application/edit/degree/additional-university/#/)=application/edit_degree_additional-university_details
document(path\=/cs/2015/suse/application/edit/degree/additional-university/##/)=application/edit_degree_additional-university_details
document(path\=/cs/2015/suse/application/edit/degree/additional-university/###/)=application/edit_degree_additional-university_details
document(path\=/cs/2015/suse/application/edit/degree/degree/)=application/edit_degree
document(path\=/cs/2015/suse/application/edit/degree/done/)=application/edit_degree_done
document(path\=/cs/2015/suse/application/edit/degree/eu/)=application/edit_degree_eu
document(path\=/cs/2015/suse/application/edit/degree/university/)=application/edit_degree_university
document(path\=/cs/2015/suse/application/edit/english/$)=application/edit_english
document(path\=/cs/2015/suse/application/edit/english/english/)=application/edit_english
document(path\=/cs/2015/suse/application/edit/english/en-nat/)=application/edit_english_en-nat
document(path\=/cs/2015/suse/application/edit/english/en-other/)=application/edit_english_en-other
document(path\=/cs/2015/suse/application/edit/english/en-test/)=application/edit_english_en-test
document(path\=/cs/2015/suse/application/edit/german/$)=application/edit_german
document(path\=/cs/2015/suse/application/edit/german/de-nat/)=application/edit_german_de-nat
document(path\=/cs/2015/suse/application/edit/german/de-other/)=application/edit_german_de-other
document(path\=/cs/2015/suse/application/edit/german/de-test/)=application/edit_german_de-test
document(path\=/cs/2015/suse/application/edit/german/german/)=application/edit_german
document(path\=/cs/2015/suse/application/edit/gsv/$)=application/edit_gsv
document(path\=/cs/2015/suse/application/edit/gsv/gsv-document/)=application/edit_gsv_gsv-document
document(path\=/cs/2015/suse/application/edit/personal)=application/edit_personal
document(path\=/cs/2015/suse/application/edit/st/$)=application/edit_st
document(path\=/cs/2015/suse/application/edit/st/special-treatment/$)=application/edit_st_special-treatment
document(path\=/cs/2015/suse/application/edit/st/special-treatment-document/$)=application/edit_st_special-treatment-document
document(path\=/cs/2015/suse/application/file/)=application/file
document(path\=/cs/2015/suse/application/withdraw/)=application/withdraw
document(path\=/cs/2015/suse/auth/user/)=auth/user
document(path\=/cs/2015/suse/imprint/)=imprint
document(path\=/cs/2015/suse/review/$)=CLEAR_QUERY,review/overview
document(path\=/cs/2015/suse/review/#/$)=review/details
document(path\=/cs/2015/suse/review/##/$)=review/details
document(path\=/cs/2015/suse/review/###/$)=review/details
document(path\=/cs/2015/suse/review/####/$)=review/details
document(path\=/cs/2015/suse/review/#/view$)=review/details_view
document(path\=/cs/2015/suse/review/##/view$)=review/details_view
document(path\=/cs/2015/suse/review/###/view$)=review/details_view
document(path\=/cs/2015/suse/review/####/view$)=review/details_view
document(path\=/cs/2015/suse/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs/2015/suse/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs/2015/suse/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs/2015/suse/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/cs/2015/suse/review/#/commit_history$)=review/details_history
document(path\=/cs/2015/suse/review/##/commit_history$)=review/details_history
document(path\=/cs/2015/suse/review/###/commit_history$)=review/details_history
document(path\=/cs/2015/suse/review/####/commit_history$)=review/details_history
document(path\=/cs/2015/suse/review/#/interview$)=review/details_interview
document(path\=/cs/2015/suse/review/##/interview$)=review/details_interview
document(path\=/cs/2015/suse/review/###/interview$)=review/details_interview
document(path\=/cs/2015/suse/review/####/interview$)=review/details_interview
document(path\=/cs/2015/suse/review/admin/$)=review/admin
document(path\=/cs/2015/suse/review/admin/#$)=review/admin_details
document(path\=/cs/2015/suse/review/admin/##$)=review/admin_details
document(path\=/cs/2015/suse/review/admin/###$)=review/admin_details
document(path\=/cs/2015/suse/review/admin/add)=review/admin_add
document(path\=/cs/2015/suse/review/admin/deadline)=review/admin_deadline
document(path\=/cs/2015/suse/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/cs/2015/suse/review/distribute/)=review/distribute
document(path\=/cs/2015/suse/review/export/)=review/export
document(path\=/cs/2015/suse/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/cs/2015/suse/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/cs/2015/suse/review/statistics)=review/statistics
document(path\=/cs/2015/suse/wizard/$)=wizard
document(path\=/cs/2015/suse/wizard/additional-degree/$)=wizard/additional-degree
document(path\=/cs/2015/suse/wizard/additional-degree/#/$)=wizard/additional-degree_details
document(path\=/cs/2015/suse/wizard/additional-degree/##/$)=wizard/additional-degree_details
document(path\=/cs/2015/suse/wizard/additional-university/#/$)=wizard/additional-university_details
document(path\=/cs/2015/suse/wizard/additional-university/##/$)=wizard/additional-university_details
document(path\=/cs/2015/suse/wizard/bologna/)=wizard/bologna
document(path\=/cs/2015/suse/wizard/degree/)=wizard/degree
document(path\=/cs/2015/suse/wizard/de-nat/)=wizard/de-nat
document(path\=/cs/2015/suse/wizard/de-other/)=wizard/de-other
document(path\=/cs/2015/suse/wizard/de-test/)=wizard/de-test
document(path\=/cs/2015/suse/wizard/english/)=wizard/english
document(path\=/cs/2015/suse/wizard/en-nat/)=wizard/en-nat
document(path\=/cs/2015/suse/wizard/en-other/)=wizard/en-other
document(path\=/cs/2015/suse/wizard/en-test/)=wizard/en-test
document(path\=/cs/2015/suse/wizard/eu/)=wizard/eu
document(path\=/cs/2015/suse/wizard/german/)=wizard/german
document(path\=/cs/2015/suse/wizard/gsv/)=wizard/gsv
document(path\=/cs/2015/suse/wizard/gsv-document/)=wizard/gsv-document
document(path\=/cs/2015/suse/wizard/special-treatment/)=wizard/special-treatment
document(path\=/cs/2015/suse/wizard/special-treatment-document/)=wizard/special-treatment-document
document(path\=/cs/2015/suse/wizard/personal/)=wizard/personal
document(path\=/cs/2015/suse/wizard/university/)=wizard/university

document(path\=/csnb/2015/suse/a$)=invalid-page
document(path\=/csnb/2015/suse/accounts$)=accounts
document(path\=/csnb/2015/suse/accounts/activate/########################################/$)=accounts/activate
document(path\=/csnb/2015/suse/accounts/activate/################################################################################/$)=account\
    s/activate
document(path\=/csnb/2015/suse/accounts/login/$)=CLEAR_QUERY,accounts/login
document(path\=/csnb/2015/suse/accounts/login/#Gottingen%20Uni#)=invalid-page
document(path\=/csnb/2015/suse/accounts/password/change/)=accounts/password_change
document(path\=/csnb/2015/suse/accounts/password/reset/$)=accounts/password_reset
document(path\=/csnb/2015/suse/accounts/password/reset/confirm/###########################/$)=accounts/password_reset_confirm
document(path\=/csnb/2015/suse/accounts/password/reset/confirm/##########################/$)=accounts/password_reset_confirm
document(path\=/csnb/2015/suse/accounts/register/$)=accounts/register
document(path\=/csnb/2015/suse/accounts/register/complete/)=accounts/register_complete
document(path\=/csnb/2015/suse/admin)=invalid-page
document(path\=/csnb/2015/suse/application/$)=application
document(path\=/csnb/2015/suse/application/edit/contact)=application/edit_contact
document(path\=/csnb/2015/suse/application/edit/degree/$)=application/edit_degree
document(path\=/csnb/2015/suse/application/edit/degree/additional-degree/$)=application/edit_degree_additional-degree
document(path\=/csnb/2015/suse/application/edit/degree/additional-degree/#/)=application/edit_degree_additional-degree_details
document(path\=/csnb/2015/suse/application/edit/degree/additional-degree/##/)=application/edit_degree_additional-degree_details
document(path\=/csnb/2015/suse/application/edit/degree/additional-degree/###/)=application/edit_degree_additional-degree_details
document(path\=/csnb/2015/suse/application/edit/degree/additional-university/#/)=application/edit_degree_additional-university_details
document(path\=/csnb/2015/suse/application/edit/degree/additional-university/##/)=application/edit_degree_additional-university_details
document(path\=/csnb/2015/suse/application/edit/degree/additional-university/###/)=application/edit_degree_additional-university_details
document(path\=/csnb/2015/suse/application/edit/degree/degree/)=application/edit_degree
document(path\=/csnb/2015/suse/application/edit/degree/eu/)=application/edit_degree_eu
document(path\=/csnb/2015/suse/application/edit/degree/university/)=application/edit_degree_university
document(path\=/csnb/2015/suse/application/edit/english/$)=application/edit_english
document(path\=/csnb/2015/suse/application/edit/english/english/)=application/edit_english
document(path\=/csnb/2015/suse/application/edit/english/en-nat/)=application/edit_english_en-nat
document(path\=/csnb/2015/suse/application/edit/english/en-other/)=application/edit_english_en-other
document(path\=/csnb/2015/suse/application/edit/english/en-test/)=application/edit_english_en-test
document(path\=/csnb/2015/suse/application/edit/german/$)=application/edit_german
document(path\=/csnb/2015/suse/application/edit/german/de-nat/)=application/edit_german_de-nat
document(path\=/csnb/2015/suse/application/edit/german/de-other/)=application/edit_german_de-other
document(path\=/csnb/2015/suse/application/edit/german/de-test/)=application/edit_german_de-test
document(path\=/csnb/2015/suse/application/edit/german/german/)=application/edit_german
document(path\=/csnb/2015/suse/application/edit/gsv/$)=application/edit_gsv
document(path\=/csnb/2015/suse/application/edit/gsv/gsv/$)=application/edit_gsv
document(path\=/csnb/2015/suse/application/edit/gsv/gsv-document/)=application/edit_gsv_gsv-document
document(path\=/csnb/2015/suse/application/edit/personal)=application/edit_personal
document(path\=/csnb/2015/suse/application/edit/st/$)=application/edit_st
document(path\=/csnb/2015/suse/application/edit/st/special-treatment/$)=application/edit_st_special-treatment
document(path\=/csnb/2015/suse/application/edit/st/special-treatment-document/$)=application/edit_st_special-treatment-document
document(path\=/csnb/2015/suse/application/file/)=application/file
document(path\=/csnb/2015/suse/application/submit/)=application/submit
document(path\=/csnb/2015/suse/application/withdraw/)=application/withdraw
document(path\=/csnb/2015/suse/auth/user/)=auth/user
document(path\=/csnb/2015/suse/imprint/)=imprint
document(path\=/csnb/2015/suse/login$)=invalid-page
document(path\=/csnb/2015/suse/r$)=invalid-page
document(path\=/csnb/2015/suse/review/$)=CLEAR_QUERY,review/overview
document(path\=/csnb/2015/suse/review/#/$)=review/details
document(path\=/csnb/2015/suse/review/##/$)=review/details
document(path\=/csnb/2015/suse/review/###/$)=review/details
document(path\=/csnb/2015/suse/review/####/$)=review/details
document(path\=/csnb/2015/suse/review/#/view$)=review/details_view
document(path\=/csnb/2015/suse/review/##/view$)=review/details_view
document(path\=/csnb/2015/suse/review/###/view$)=review/details_view
document(path\=/csnb/2015/suse/review/####/view$)=review/details_view
document(path\=/csnb/2015/suse/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/csnb/2015/suse/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/csnb/2015/suse/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/csnb/2015/suse/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/csnb/2015/suse/review/#/commit_history$)=review/details_history
document(path\=/csnb/2015/suse/review/##/commit_history$)=review/details_history
document(path\=/csnb/2015/suse/review/###/commit_history$)=review/details_history
document(path\=/csnb/2015/suse/review/####/commit_history$)=review/details_history
document(path\=/csnb/2015/suse/review/#/interview$)=review/details_interview
document(path\=/csnb/2015/suse/review/##/interview$)=review/details_interview
document(path\=/csnb/2015/suse/review/###/interview$)=review/details_interview
document(path\=/csnb/2015/suse/review/####/interview$)=review/details_interview
document(path\=/csnb/2015/suse/review/admin/$)=review/admin
document(path\=/csnb/2015/suse/review/admin/#$)=review/admin_details
document(path\=/csnb/2015/suse/review/admin/##$)=review/admin_details
document(path\=/csnb/2015/suse/review/admin/###$)=review/admin_details
document(path\=/csnb/2015/suse/review/admin/add)=review/admin_add
document(path\=/csnb/2015/suse/review/admin/deadline)=review/admin_deadline
document(path\=/csnb/2015/suse/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/csnb/2015/suse/review/distribute/)=review/distribute
document(path\=/csnb/2015/suse/review/export/)=review/export
document(path\=/csnb/2015/suse/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/csnb/2015/suse/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/csnb/2015/suse/review/statistics)=review/statistics
document(path\=/csnb/2015/suse/wizard/$)=wizard
document(path\=/csnb/2015/suse/wizard/additional-degree/$)=wizard/additional-degree
document(path\=/csnb/2015/suse/wizard/additional-degree/#/$)=wizard/additional-degree_details
document(path\=/csnb/2015/suse/wizard/additional-degree/##/$)=wizard/additional-degree_details
document(path\=/csnb/2015/suse/wizard/additional-university/#/$)=wizard/additional-university_details
document(path\=/csnb/2015/suse/wizard/additional-university/##/$)=wizard/additional-university_details
document(path\=/csnb/2015/suse/wizard/bologna/)=wizard/bologna
document(path\=/csnb/2015/suse/wizard/degree/)=wizard/degree
document(path\=/csnb/2015/suse/wizard/de-nat/)=wizard/de-nat
document(path\=/csnb/2015/suse/wizard/de-other/)=wizard/de-other
document(path\=/csnb/2015/suse/wizard/de-test/)=wizard/de-test
document(path\=/csnb/2015/suse/wizard/english/)=wizard/english
document(path\=/csnb/2015/suse/wizard/en-nat/)=wizard/en-nat
document(path\=/csnb/2015/suse/wizard/en-other/)=wizard/en-other
document(path\=/csnb/2015/suse/wizard/en-test/)=wizard/en-test
document(path\=/csnb/2015/suse/wizard/eu/)=wizard/eu
document(path\=/csnb/2015/suse/wizard/german/)=wizard/german
document(path\=/csnb/2015/suse/wizard/gsv/)=wizard/gsv
document(path\=/csnb/2015/suse/wizard/gsv-document/)=wizard/gsv-document
document(path\=/csnb/2015/suse/wizard/special-treatment/)=wizard/special-treatment
document(path\=/csnb/2015/suse/wizard/special-treatment-document/)=wizard/special-treatment-document
document(path\=/csnb/2015/suse/wizard/personal/)=wizard/personal
document(path\=/csnb/2015/suse/wizard/university/)=wizard/university

document(path\=/itis/201$)=invalid-page
document(path\=/itis/2013/suse$)=invalid-page

document(path\=/itis/2013/wise/$)=invalid-page
document(path\=/itis/2013/wise/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2013/wise/accounts/password/reset/)=accounts/password_reset
document(path\=/itis/2013/wise/review/)=review/overview

document(path\=/itis2014/$)=/
document(path\=/itis/2014$)=/
document(path\=/itis/2014/$)=/

document(path\=/itis/2014/application/)=application

document(path\=/itis/2014/sose/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/sose/accounts/password/change/)=accounts/password_change
document(path\=/itis/2014/sose/accounts/password/reset/)=accounts/password_reset
document(path\=/itis/2014/sose/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/2014/sose/review/#/$)=review/details
document(path\=/itis/2014/sose/review/##/$)=review/details
document(path\=/itis/2014/sose/review/###/$)=review/details
document(path\=/itis/2014/sose/review/####/$)=review/details
document(path\=/itis/2014/sose/review/#/view$)=review/details_view
document(path\=/itis/2014/sose/review/##/view$)=review/details_view
document(path\=/itis/2014/sose/review/###/view$)=review/details_view
document(path\=/itis/2014/sose/review/####/view$)=review/details_view
document(path\=/itis/2014/sose/review/#/commit_history$)=review/details_history
document(path\=/itis/2014/sose/review/##/commit_history$)=review/details_history
document(path\=/itis/2014/sose/review/###/commit_history$)=review/details_history
document(path\=/itis/2014/sose/review/####/commit_history$)=review/details_history
document(path\=/itis/2014/sose/review/admin/)=review/admin
document(path\=/itis/2014/sose/review/finalized/)=CLEAR_QUERY,review/finalized
document(path\=/itis/2014/sose/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/itis/2014/sose/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/sose/review/notified/)=CLEAR_QUERY,review/notified

document(path\=/itis/2014/suse$)=/
document(path\=/itis/2014/suse/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/suse/application)=application
document(path\=/itis/2014/suse/review$)=review/overview
document(path\=/itis/2014/suse/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/2014/suse/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/wise/accounts$)=accounts
document(path\=/itis/2014/wise/accounts/$)=accounts
document(path\=/itis/2014/wise/accounts//$)=accounts
document(path\=/itis/2014/wise/accounts/activate/############################################/)=accounts/activate
document(path\=/itis/2014/wise/accounts/active/##########################################/)=accounts/activate
document(path\=/itis/2014/wise/accounts/active/###########################################/)=accounts/activate
document(path\=/itis/2014/wise/accounts/active/############################################/)=accounts/activate
document(path\=/itis/2014/wise/accounts/create/)=CLEAR_QUERY,accounts/create
document(path\=/itis/2014/wise/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/wise/accounts/password/change/)=accounts/password_change
document(path\=/itis/2014/wise/accounts/password/reset/$)=accounts/password_reset
document(path\=/itis/2014/wise/accounts/password/reset/confirm/############################/)=accounts/password_reset_confirm
document(path\=/itis/2014/wise/accounts/password/reset/confirm/#############################/)=accounts/password_reset_confirm
document(path\=/itis/2014/wise/accounts/register/$)=accounts/register
document(path\=/itis/2014/wise/accounts/register/complete/)=accounts/register_complete
document(path\=/itis/2014/wise/accounts/signup/$)=accounts/register
document(path\=/itis/2014/wise/accountss/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/wise/accounts/user)=invalid-page
document(path\=/itis/2014/wise/application/$)=application
document(path\=/itis/2014/wise/application//)=application
document(path\=/itis/2014/wise/application/%5C)=application
document(path\=/itis/2014/wise/application/edit/additionalinformation/)=application/edit_additionalinformation
document(path\=/itis/2014/wise/application/edit/contact)=application/edit_contact
document(path\=/itis/2014/wise/application/edit/degree/$)=application/edit_degree
document(path\=/itis/2014/wise/application/edit/degree/degree/)=application/edit_degree
document(path\=/itis/2014/wise/application/edit/degree/university/)=application/edit_degree_university
document(path\=/itis/2014/wise/application/edit/financialarrangement/)=application/edit_financialarrangement
document(path\=/itis/2014/wise/application/edit/language/$)=application/edit_english
document(path\=/itis/2014/wise/application/edit/language/english/)=application/edit_english
document(path\=/itis/2014/wise/application/edit/language/en-nat/)=application/edit_english_en-nat
document(path\=/itis/2014/wise/application/edit/language/en-study-work/)=application/edit_english_en-study-work
document(path\=/itis/2014/wise/application/edit/language/en-study-work-files/)=application/edit_english_en-study-work-files
document(path\=/itis/2014/wise/application/edit/language/en-test/)=application/edit_english_en-test
document(path\=/itis/2014/wise/application/edit/personal)=application/edit_personal
document(path\=/itis/2014/wise/application/edit/researchfocus/)=application/edit_researchfocus
document(path\=/itis/2014/wise/application/edit/study/$)=application/edit_study
document(path\=/itis/2014/wise/application/edit/study/further-degree/$)=application/edit_degree_additional-degree
document(path\=/itis/2014/wise/application/edit/study/further-degree/#/$)=application/edit_degree_additional-degree_details
document(path\=/itis/2014/wise/application/edit/study/further-degree/##/$)=application/edit_degree_additional-degree_details
document(path\=/itis/2014/wise/application/edit/study/further-university/#/$)=application/edit_degree_additional-university_details
document(path\=/itis/2014/wise/application/edit/study/further-university/##/$)=application/edit_degree_additional-university_details
document(path\=/itis/2014/wise/application/edit/work/$)=application/edit_work
document(path\=/itis/2014/wise/application/edit/work/opt-work-exp/$)=application/edit_work
document(path\=/itis/2014/wise/application/edit/work/opt-work-exp/#/$)=application/edit_work_details
document(path\=/itis/2014/wise/application/edit/work/opt-work-exp/##/$)=application/edit_work_details
document(path\=/itis/2014/wise/application/file/)=application/file
document(path\=/itis/2014/wise/applicationuniassist)=invalid-page
document(path\=/itis/2014/wise/application/withdraw/)=application/withdraw
document(path\=/itis/2014/wise/imprint/)=imprint
document(path\=/itis/2014/wise/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/2014/wise/review/#/$)=review/details
document(path\=/itis/2014/wise/review/##/$)=review/details
document(path\=/itis/2014/wise/review/###/$)=review/details
document(path\=/itis/2014/wise/review/####/$)=review/details
document(path\=/itis/2014/wise/review/#/view$)=review/details_view
document(path\=/itis/2014/wise/review/##/view$)=review/details_view
document(path\=/itis/2014/wise/review/###/view$)=review/details_view
document(path\=/itis/2014/wise/review/####/view$)=review/details_view
document(path\=/itis/2014/wise/review/#/commit_history$)=review/details_history
document(path\=/itis/2014/wise/review/##/commit_history$)=review/details_history
document(path\=/itis/2014/wise/review/###/commit_history$)=review/details_history
document(path\=/itis/2014/wise/review/####/commit_history$)=review/details_history
document(path\=/itis/2014/wise/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2014/wise/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2014/wise/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2014/wise/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2014/wise/review/admin/$)=review/admin
document(path\=/itis/2014/wise/review/admin/#$)=review/admin_details
document(path\=/itis/2014/wise/review/admin/##$)=review/admin_details
document(path\=/itis/2014/wise/review/admin/###$)=review/admin_details
document(path\=/itis/2014/wise/review/admin/deadline)=review/admin_deadline
document(path\=/itis/2014/wise/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/itis/2014/wise/review/distribute/)=review/distribute
document(path\=/itis/2014/wise/review/finalized/)=CLEAR_QUERY,review/finalized
document(path\=/itis/2014/wise/review/file/701)=invalid-page
document(path\=/itis/2014/wise/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/itis/2014/wise/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2014/wise/review/notified/)=CLEAR_QUERY,review/notified
document(path\=/itis/2014/wise/review/statistics/)=review/statistics
document(path\=/itis/2014/wise/review/universities/)=review/universities
document(path\=/itis/2014/wise/rewiew)=invalid-page
document(path\=/itis/2014/wise/wizard/$)=wizard
document(path\=/itis/2014/wise/wizard/additional/)=wizard/additional
document(path\=/itis/2014/wise/wizard/degree/)=wizard/degree
document(path\=/itis/2014/wise/wizard/english/)=wizard/english
document(path\=/itis/2014/wise/wizard/en-nat/)=wizard/en-nat
document(path\=/itis/2014/wise/wizard/en-study-work/)=wizard/en-study-work
document(path\=/itis/2014/wise/wizard/en-test/)=wizard/en-test
document(path\=/itis/2014/wise/wizard/finance/)=wizard/finance
document(path\=/itis/2014/wise/wizard/further-degree/$)=wizard/additional-degree
document(path\=/itis/2014/wise/wizard/further-degree/#/$)=wizard/additional-degree_details
document(path\=/itis/2014/wise/wizard/further-degree/##/$)=wizard/additional-degree_details
document(path\=/itis/2014/wise/wizard/further-university/#/$)=wizard/additional-university_details
document(path\=/itis/2014/wise/wizard/further-university/##/$)=wizard/additional-university_details
document(path\=/itis/2014/wise/wizard/opt-work-exp/$)=wizard/opt-work-exp
document(path\=/itis/2014/wise/wizard/opt-work-exp/#/$)=wizard/opt-work-exp_details
document(path\=/itis/2014/wise/wizard/opt-work-exp/##/$)=wizard/opt-work-exp_details
document(path\=/itis/2014/wise/wizard/personal/)=wizard/personal
document(path\=/itis/2014/wise/wizard/research-focus/)=wizard/research-focus
document(path\=/itis/2014/wise/wizard/university/)=wizard/university

document(path\=/itis/2015$)=/
document(path\=/itis/2015/$)=/

document(path\=/itis/2015/sose/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/sose/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/2015/sose/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/sose/review/notified/)=CLEAR_QUERY,review/notified

document(path\=/itis/2015/suse/$)=/
document(path\=/itis/2015/suse/accounts$)=accounts
document(path\=/itis/2015/suse/accounts/$)=accounts
document(path\=/itis/2015/suse/accounts//$)=accounts
document(path\=/itis/2015/suse/accounts/activate/#######################################/)=accounts/activate
document(path\=/itis/2015/suse/accounts/activate/########################################/)=accounts/activate
document(path\=/itis/2015/suse/accounts/lcreate)=invalid-page
document(path\=/itis/2015/suse/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/suse/accounts/password/change/)=accounts/password_change
document(path\=/itis/2015/suse/accounts/password/reset/$)=accounts/password_reset
document(path\=/itis/2015/suse/accounts/password/reset/confirm/##########################/)=accounts/password_reset_confirm
document(path\=/itis/2015/suse/accounts/password/reset/confirm/###########################/)=accounts/password_reset_confirm
document(path\=/itis/2015/suse/accounts/register/$)=accounts/register
document(path\=/itis/2015/suse/accounts/register/complete/)=accounts/register_complete
document(path\=/itis/2015/suse/accounts/signup/$)=accounts/register
document(path\=/itis/2015/suse/accountss/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/suse/accounts/user)=invalid-page
document(path\=/itis/2015/suse/application/$)=application
document(path\=/itis/2015/suse/appplication/$)=invalid-page
document(path\=/itis/2015/suse/application/edit/additionalinformation/)=application/edit_additionalinformation
document(path\=/itis/2015/suse/application/edit/contact)=application/edit_contact
document(path\=/itis/2015/suse/application/edit/degree/$)=application/edit_degree
document(path\=/itis/2015/suse/application/edit/degree/degree/)=application/edit_degree
document(path\=/itis/2015/suse/application/edit/degree/university/)=application/edit_degree_university
document(path\=/itis/2015/suse/application/edit/financialarrangement/)=application/edit_financialarrangement
document(path\=/itis/2015/suse/application/edit/language/$)=application/edit_english
document(path\=/itis/2015/suse/application/edit/language/english/)=application/edit_english
document(path\=/itis/2015/suse/application/edit/language/en-nat/)=application/edit_english_en-nat
document(path\=/itis/2015/suse/application/edit/language/en-study-work/)=application/edit_english_en-study-work
document(path\=/itis/2015/suse/application/edit/language/en-study-work-files/)=application/edit_english_en-study-work-files
document(path\=/itis/2015/suse/application/edit/language/en-test/)=application/edit_english_en-test
document(path\=/itis/2015/suse/application/edit/personal)=application/edit_personal
document(path\=/itis/2015/suse/application/edit/researchfocus/)=application/edit_researchfocus
document(path\=/itis/2015/suse/application/edit/study/$)=application/edit_study
document(path\=/itis/2015/suse/application/edit/study/further-degree/$)=application/edit_degree_additional-degree
document(path\=/itis/2015/suse/application/edit/study/further-degree/#/$)=application/edit_degree_additional-degree_details
document(path\=/itis/2015/suse/application/edit/study/further-degree/##/$)=application/edit_degree_additional-degree_details
document(path\=/itis/2015/suse/application/edit/study/further-university/#/$)=application/edit_degree_additional-university_details
document(path\=/itis/2015/suse/application/edit/study/further-university/##/$)=application/edit_degree_additional-university_details
document(path\=/itis/2015/suse/application/edit/work/$)=application/edit_work
document(path\=/itis/2015/suse/application/edit/work/opt-work-exp/$)=application/edit_work
document(path\=/itis/2015/suse/application/edit/work/opt-work-exp/#/$)=application/edit_work_details
document(path\=/itis/2015/suse/application/edit/work/opt-work-exp/##/$)=application/edit_work_details
document(path\=/itis/2015/suse/application/result)=invalid-page
document(path\=/itis/2015/suse/application/submit/)=application/submit
document(path\=/itis/2015/suse/application/withdraw/)=application/withdraw
document(path\=/itis/2015/suse/imprint/)=imprint
document(path\=/itis/2015/suse/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/2015/suse/review/#/$)=review/details
document(path\=/itis/2015/suse/review/##/$)=review/details
document(path\=/itis/2015/suse/review/###/$)=review/details
document(path\=/itis/2015/suse/review/####/$)=review/details
document(path\=/itis/2015/suse/review/#/view$)=review/details_view
document(path\=/itis/2015/suse/review/##/view$)=review/details_view
document(path\=/itis/2015/suse/review/###/view$)=review/details_view
document(path\=/itis/2015/suse/review/####/view$)=review/details_view
document(path\=/itis/2015/suse/review/#/commit_history$)=review/details_history
document(path\=/itis/2015/suse/review/##/commit_history$)=review/details_history
document(path\=/itis/2015/suse/review/###/commit_history$)=review/details_history
document(path\=/itis/2015/suse/review/####/commit_history$)=review/details_history
document(path\=/itis/2015/suse/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/suse/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/suse/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/suse/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/suse/review/admin/$)=review/admin
document(path\=/itis/2015/suse/review/admin/deadline)=review/admin_deadline
document(path\=/itis/2015/suse/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/itis/2015/suse/review/distribute/)=review/distribute
document(path\=/itis/2015/suse/review/finalized/)=CLEAR_QUERY,review/finalized
document(path\=/itis/2015/suse/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/itis/2015/suse/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/suse/review/notified/)=CLEAR_QUERY,review/notified
document(path\=/itis/2015/suse/review/universities/)=review/universities
document(path\=/itis/2015/suse/wizard/$)=wizard
document(path\=/itis/2015/suse/wizard/additional/)=wizard/additional
document(path\=/itis/2015/suse/wizard/degree/)=wizard/degree
document(path\=/itis/2015/suse/wizard/english/)=wizard/english
document(path\=/itis/2015/suse/wizard/en-nat/)=wizard/en-nat
document(path\=/itis/2015/suse/wizard/en-study-work/)=wizard/en-study-work
document(path\=/itis/2015/suse/wizard/en-test/)=wizard/en-test
document(path\=/itis/2015/suse/wizard/finance/)=wizard/finance
document(path\=/itis/2015/suse/wizard/further-degree/$)=wizard/additional-degree
document(path\=/itis/2015/suse/wizard/further-degree/#/$)=wizard/additional-degree_details
document(path\=/itis/2015/suse/wizard/further-degree/##/$)=wizard/additional-degree_details
document(path\=/itis/2015/suse/wizard/further-university/#/$)=wizard/additional-university_details
document(path\=/itis/2015/suse/wizard/further-university/##/$)=wizard/additional-university_details
document(path\=/itis/2015/suse/wizard/opt-work-exp/$)=wizard/opt-work-exp
document(path\=/itis/2015/suse/wizard/opt-work-exp/#/$)=wizard/opt-work-exp_details
document(path\=/itis/2015/suse/wizard/opt-work-exp/##/$)=wizard/opt-work-exp_details
document(path\=/itis/2015/suse/wizard/personal/)=wizard/personal
document(path\=/itis/2015/suse/wizard/research-focus/)=wizard/research-focus
document(path\=/itis/2015/suse/wizard/university/)=wizard/university

document(path\=/itis/2015/wise/$)=/
document(path\=/itis/2015/wise/account/login$)=invalid-page
document(path\=/itis/2015/wiseaccounts/$)=invalid-page
document(path\=/itis/2015/wise/accounts$)=accounts
document(path\=/itis/2015/wise/accounts/$)=accounts
document(path\=/itis/2015/wise/accounts//$)=accounts
document(path\=/itis/2015/wise/accounts/activate/########################################/)=accounts/activate
document(path\=/itis/2015/wise/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/wise/accounts/login/#)=invalid-page
document(path\=/itis/2015/wise/accounts/new-user/)=invalid-page
document(path\=/itis/2015/wise/accounts/password/change/)=accounts/password_change
document(path\=/itis/2015/wise/accounts/password/reset/$)=accounts/password_reset
document(path\=/itis/2015/wise/accounts/password/reset/confirm/##########################/)=accounts/password_reset_confirm
document(path\=/itis/2015/wise/accounts/password/reset/confirm/###########################/)=accounts/password_reset_confirm
document(path\=/itis/2015/wise/accounts/password/reset/confirm/############################/)=accounts/password_reset_confirm
document(path\=/itis/2015/wise/accounts/register/$)=accounts/register
document(path\=/itis/2015/wise/accounts/r/egister$)=invalid-page
document(path\=/itis/2015/wise/accounts/register/complete/)=accounts/register_complete
document(path\=/itis/2015/wise/application/$)=application
document(path\=/itis/2015/wise/application/edit/additionalinformation/)=application/edit_additionalinformation
document(path\=/itis/2015/wise/application/edit/contact)=application/edit_contact
document(path\=/itis/2015/wise/application/edit/degree/$)=application/edit_degree
document(path\=/itis/2015/wise/application/edit/degree/degree/)=application/edit_degree
document(path\=/itis/2015/wise/application/edit/degree/university/)=application/edit_degree_university
document(path\=/itis/2015/wise/application/edit/financialarrangement/)=application/edit_financialarrangement
document(path\=/itis/2015/wise/application/edit/language/$)=application/edit_english
document(path\=/itis/2015/wise/application/edit/language/english/)=application/edit_english
document(path\=/itis/2015/wise/application/edit/language/en-nat/)=application/edit_english_en-nat
document(path\=/itis/2015/wise/application/edit/language/en-study-work/)=application/edit_english_en-study-work
document(path\=/itis/2015/wise/application/edit/language/en-study-work-files/)=application/edit_english_en-study-work-files
document(path\=/itis/2015/wise/application/edit/language/en-test/)=application/edit_english_en-test
document(path\=/itis/2015/wise/application/edit/personal)=application/edit_personal
document(path\=/itis/2015/wise/application/edit/researchfocus/)=application/edit_researchfocus
document(path\=/itis/2015/wise/application/edit/study/$)=application/edit_study
document(path\=/itis/2015/wise/application/edit/study/further-degree/$)=application/edit_degree_additional-degree
document(path\=/itis/2015/wise/application/edit/study/further-degree/#/$)=application/edit_degree_additional-degree_details
document(path\=/itis/2015/wise/application/edit/study/further-degree/##/$)=application/edit_degree_additional-degree_details
document(path\=/itis/2015/wise/application/edit/study/further-university/#/$)=application/edit_degree_additional-university_details
document(path\=/itis/2015/wise/application/edit/study/further-university/##/$)=application/edit_degree_additional-university_details
document(path\=/itis/2015/wise/application/edit/work/$)=application/edit_work
document(path\=/itis/2015/wise/application/edit/work/opt-work-exp/$)=application/edit_work
document(path\=/itis/2015/wise/application/edit/work/opt-work-exp/#/$)=application/edit_work_details
document(path\=/itis/2015/wise/application/edit/work/opt-work-exp/##/$)=application/edit_work_details
document(path\=/itis/2015/wise/application/submit/)=application/submit
document(path\=/itis/2015/wise/application/withdraw/)=application/withdraw
document(path\=/itis/2015/wise/imprint/)=imprint
document(path\=/itis/2015/wise/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/2015/wise/review/#/$)=review/details
document(path\=/itis/2015/wise/review/##/$)=review/details
document(path\=/itis/2015/wise/review/###/$)=review/details
document(path\=/itis/2015/wise/review/####/$)=review/details
document(path\=/itis/2015/wise/review/#/view$)=review/details_view
document(path\=/itis/2015/wise/review/##/view$)=review/details_view
document(path\=/itis/2015/wise/review/###/view$)=review/details_view
document(path\=/itis/2015/wise/review/####/view$)=review/details_view
document(path\=/itis/2015/wise/review/#/commit_history$)=review/details_history
document(path\=/itis/2015/wise/review/##/commit_history$)=review/details_history
document(path\=/itis/2015/wise/review/###/commit_history$)=review/details_history
document(path\=/itis/2015/wise/review/####/commit_history$)=review/details_history
document(path\=/itis/2015/wise/review/#/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/wise/review/##/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/wise/review/###/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/wise/review/####/same_university$)=CLEAR_QUERY,review/details_same_university
document(path\=/itis/2015/wise/review/admin/$)=review/admin
document(path\=/itis/2015/wise/review/admin/deadline)=review/admin_deadline
document(path\=/itis/2015/wise/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/itis/2015/wise/review/distribute/)=review/distribute
document(path\=/itis/2015/wise/review/finalized/)=CLEAR_QUERY,review/finalized
document(path\=/itis/2015/wise/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/itis/2015/wise/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/2015/wise/review/notified/)=CLEAR_QUERY,review/notified
document(path\=/itis/2015/wise/review/universities/)=review/universities
document(path\=/itis/2015/wise/wizard/$)=wizard
document(path\=/itis/2015/wise/wizard/additional/)=wizard/additional
document(path\=/itis/2015/wise/wizard/degree/)=wizard/degree
document(path\=/itis/2015/wise/wizard/english/)=wizard/english
document(path\=/itis/2015/wise/wizard/en-nat/)=wizard/en-nat
document(path\=/itis/2015/wise/wizard/en-study-work/)=wizard/en-study-work
document(path\=/itis/2015/wise/wizard/en-test/)=wizard/en-test
document(path\=/itis/2015/wise/wizard/finance/)=wizard/finance
document(path\=/itis/2015/wise/wizard/further-degree/$)=wizard/additional-degree
document(path\=/itis/2015/wise/wizard/further-degree/#/$)=wizard/additional-degree_details
document(path\=/itis/2015/wise/wizard/further-degree/##/$)=wizard/additional-degree_details
document(path\=/itis/2015/wise/wizard/further-university/#/$)=wizard/additional-university_details
document(path\=/itis/2015/wise/wizard/further-university/##/$)=wizard/additional-university_details
document(path\=/itis/2015/wise/wizard/opt-work-exp/$)=wizard/opt-work-exp
document(path\=/itis/2015/wise/wizard/opt-work-exp/#/$)=wizard/opt-work-exp_details
document(path\=/itis/2015/wise/wizard/opt-work-exp/##/$)=wizard/opt-work-exp_details
document(path\=/itis/2015/wise/wizard/personal/)=wizard/personal
document(path\=/itis/2015/wise/wizard/research-focus/)=wizard/research-focus
document(path\=/itis/2015/wise/wizard/university/)=wizard/university
document(path\=/itis/2015/wsse/accounts/login/)=CLEAR_QUERY,accounts/login

document(path\=/itis/2016$)=/

document(path\=/itis/2016/wise/accounts/login/)=CLEAR_QUERY,accounts/login

document(path\=/itis/accounts/$)=accounts
document(path\=/itis/accounts/activate/########################################/)=accounts/activate
document(path\=/itis/accounts/create/)=CLEAR_QUERY,accounts/create
document(path\=/itis/accounts/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/accounts/password/reset/$)=accounts/password_reset
document(path\=/itis/accounts/password/reset/confirm/###########################/)=accounts/password_reset_confirm
document(path\=/itis/application/$)=application
document(path\=/itis/application/withdraw/)=application/withdraw
document(path\=/itis/imprint/)=imprint
document(path\=/itis/master)=master
document(path\=/itis/review/$)=CLEAR_QUERY,review/overview
document(path\=/itis/review/#/$)=review/details
document(path\=/itis/review/##/$)=review/details
document(path\=/itis/review/###/$)=review/details
document(path\=/itis/review/####/$)=review/details
document(path\=/itis/review/#/view$)=review/details_view
document(path\=/itis/review/##/view$)=review/details_view
document(path\=/itis/review/###/view$)=review/details_view
document(path\=/itis/review/####/view$)=review/details_view
document(path\=/itis/review/#/commit_history$)=review/details_history
document(path\=/itis/review/##/commit_history$)=review/details_history
document(path\=/itis/review/###/commit_history$)=review/details_history
document(path\=/itis/review/####/commit_history$)=review/details_history
document(path\=/itis/review/csv/)=CLEAR_QUERY,review/csv
document(path\=/itis/review/distribute/)=review/distribute
document(path\=/itis/review/finalized/)=CLEAR_QUERY,review/finalized
document(path\=/itis/review/incomp/)=CLEAR_QUERY,review/incomp
document(path\=/itis/review/login/)=CLEAR_QUERY,accounts/login
document(path\=/itis/review/notified/)=CLEAR_QUERY,review/notified
document(path\=/itis/review/statistics)=review/statistics
document(path\=/itis/review/universities/)=review/universities

document(path\=/itis/suse/2015$)=invalid-page
document(path\=/itis/suse/2015/$)=invalid-page


########################################################################################
# create ids for those page parts, where the differences are
########################################################################################

body=

#body/div/div/div/form=

document(path\=/$)/html/body/div[0]/div[1]=content_root

document(path\=invalid-page)/html/body/div[0]/div[1]=invalid_page_content

document(path\=accounts)/html/body/div[1]=CLEAR_INDEX
document(path\=accounts)/html/body/div[0]/div[1]=content_accounts
document(path\=accounts/activate)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_activate
document(path\=accounts/create)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_create
document(path\=accounts/login)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_login
document(path\=accounts/login)/html/body/div[0]/div(htmlId\=content_accounts)/div(htmlId\=content_accounts_login)/form=
document(path\=accounts/login)/html/body/div[0]/div(htmlId\=content_accounts)/div(htmlId\=content_accounts_login)/form[0]/div[3]=login-actions
document(path\=accounts/login)/html/body/div[0]/div(htmlId\=content_accounts)/div(htmlId\=content_accounts_login)/form[0]/div[4]=login-actions
document(path\=accounts/new)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_new
document(path\=accounts/password_change)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_password_change
document(path\=accounts/password_change)/html/body/div[0]/div(htmlId\=content_accounts)/div(htmlId\=content_accounts_password_change)/form=
document(path\=accounts/password_reset$)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_password_reset
document(path\=accounts/password_reset_confirm)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_password_reset_confirm
document(path\=accounts/register$)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_register
document(path\=accounts/register$)/html/body/div[0]/div(htmlId\=content_accounts)/div(htmlId\=content_accounts_register)/form=
document(path\=accounts/register_complete)/html/body/div[0]/div(htmlId\=content_accounts)/div[0]=content_accounts_register_complete

document(path\=adm$)/html/body/div[0]/div[1]=content_adm

document(path\=application$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_contact$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_degree$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_degree_additional-degree$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_english$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_english_en-study-work$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_english_en-test$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_financialarrangement)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_gsv)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_personal)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_study)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_researchfocus)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_work$)/html/body/div[1]=CLEAR_INDEX
document(path\=application/edit_work_details)/html/body/div[1]=CLEAR_INDEX
document(path\=application)/html/body/div[0]/div[1]=content_application
document(path\=application$)/html/body/div[0]/div(htmlId\=content_application)/div[0]=content_application_overview
document(path\=application/edit_)/html/body/div[0]/div(htmlId\=content_application)/div[0]=content_application_edit
document(path\=application/edit_)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form=
document(path\=application/edit_additionalinformation)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edi\
      t)/form[0]/div[1]=content_application_edit_additionalinformation
document(path\=application/edit_contact)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div\
      [1]=content_application_edit_contact
document(path\=application/edit_degree$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div\
      [1]=content_application_edit_degree
```

```
document(path\=application/edit_degree_additional-degree$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application\
    _edit)/form[0]/div[1]=content_application_edit_degree_additional-degree
document(path\=application/edit_degree_additional-degree_details)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_appl\
    ication_edit)/form[0]/div[1]=content_application_edit_degree_additional-degree_details
document(path\=application/edit_degree_additional-university_details)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_\
    application_edit)/form[0]/div[1]=content_application_edit_degree_additional-university_details
document(path\=application/edit_degree_eu)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/d\
    iv[1]=content_application_edit_degree_eu
document(path\=application/edit_degree_university)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/f\
    orm[0]/div[1]=content_application_edit_degree_university
document(path\=application/edit_degree_done$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0\
    ]/div[1]=content_application_edit_degree_done
document(path\=application/edit_english$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/di\
    v[1]=content_application_edit_english
document(path\=application/edit_english_en-other)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/fo\
    rm[0]/div[1]=content_application_edit_english_en-other
document(path\=application/edit_english_en-test)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/for\
    m[0]/div[1]=content_application_edit_english_en-test
document(path\=application/edit_english_en-nat)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form\
    [0]/div[1]=content_application_edit_language_en-nat
document(path\=application/edit_english_en-study-work$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_ed\
    it)/form[0]/div[1]=content_application_edit_language_en-study-work
document(path\=application/edit_english_en-study-work-files)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_applicati\
    on_edit)/form[0]/div[1]=content_application_edit_language_en-study-work-files
document(path\=application/edit_financialarrangement)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit\
    )/form[0]/div[1]=content_application_edit_financialarrangement
document(path\=application/edit_german$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div\
    [1]=content_application_edit_german
document(path\=application/edit_german_de-nat)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[\
    0]/div[1]=content_application_edit_german_de-nat
document(path\=application/edit_german_de-other)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/for\
    m[0]/div[1]=content_application_edit_german_de-other
document(path\=application/edit_german_de-test)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form\
    [0]/div[1]=content_application_edit_german_de-test
document(path\=application/edit_gsv$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div[1]\
    =content_application_edit_gsv
document(path\=application/edit_gsv_gsv-document)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/fo\
    rm[0]/div[1]=content_application_edit_gsv_gsv-document
document(path\=application/edit_language$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/d\
    iv[1]=content_application_edit_language
document(path\=application/edit_personal)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/di\
    v[1]=content_application_edit_personal
document(path\=application/edit_researchfocus)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[\
    0]/div[1]=content_application_edit_researchfocus
document(path\=application/edit_st$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div[1]=\
    content_application_edit_st
document(path\=application/edit_st_special-treatment$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edi\
    t)/form[0]/div[1]=content_application_edit_st_special-treatment
document(path\=application/edit_st_special-treatment-document)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_applica\
    tion_edit)/form[0]/div[1]=content_application_edit_st_special-treatment-document
document(path\=application/edit_study)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div[1\
    ]=content_application_edit_study
document(path\=application/edit_work$)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0]/div[1\
    ]=content_application_edit_work
document(path\=application/edit_work_details)/html/body/div[0]/div(htmlId\=content_application)/div(htmlId\=content_application_edit)/form[0\
    ]/div[1]=content_application_edit_work_details
document(path\=application/file)/html/body/div[0]/div(htmlId\=content_application)/div[0]=content_application_file
document(path\=application/submit)/html/body/div[0]/div(htmlId\=content_application)/div[0]=content_application_submit
document(path\=application/withdraw)/html/body/div[0]/div(htmlId\=content_application)/div[0]=content_application_withdraw

document(path\=auth/user)/html/body/div[0]/div[1]=content_auth_user
document(path\=imprint)/html/body/div[1]=CLEAR_INDEX
document(path\=imprint)/html/body/div[0]/div[1]=content_imprint
document(path\=paths)/html/body/div[0]/div[1]=content_paths

document(path\=review)/html/body/div[0]/div[1]=content_review
document(path\=review/admin$)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_admin
document(path\=review/admin_add)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_admin_add
document(path\=review/admin_auth$)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_admin_auth
document(path\=review/admin_auth_user)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_admin_auth_user
document(path\=review/admin_deadline)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_admin_deadline
div(htmlId\=content_review_admin_deadline)/div[1]=CLEAR_INDEX
document(path\=review/admin_details)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_admin_details
document(path\=review/csv)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_csv
document(path\=review/details$)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_details
document(path\=review/details_history)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_details_history
document(path\=review/details_interview)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_details_interview
document(path\=review/details_same_university)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_details_same_university
document(path\=review/details_view)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_details_view
document(path\=review/distribute)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_distribute
document(path\=review/export)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_export
document(path\=review/finalized)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_finalized
document(path\=review/incomp)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_incomp
document(path\=review/notified)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_notified
document(path\=review/overview)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_overview
document(path\=review/statistics)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_statistics
document(path\=review/universities)/html/body/div[0]/div(htmlId\=content_review)/div[0]=content_review_universities

document(path\=wizard/additional$)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/additional-degree$)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/de-test)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/personal)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/english)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/en-test)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/en-study-work)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/eu)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/finance)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/german)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/gsv)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/research-focus)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/opt-work-exp$)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/opt-work-exp_details)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard/university)/html/body/div[1]=CLEAR_INDEX
document(path\=wizard)/html/body/div[0]/div[1]=content_wizard
div(htmlId\=content_wizard)/div[0]/form=
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[2]=content_wizard_buttons
document(path\=wizard$)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_personal
document(path\=wizard/additional$)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_additional
document(path\=wizard/additional-degree$)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_additiona\
    l-degree
document(path\=wizard/additional-degree_details)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_ad\
    ditional-degree_details
document(path\=wizard/additional-university_details)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizar\
    d_additional-university_details
document(path\=wizard/bologna)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_bologna
document(path\=wizard/degree)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_degree
document(path\=wizard/de-nat)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_de-nat
document(path\=wizard/de-other)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_de-other
document(path\=wizard/de-test)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_de-test
document(path\=wizard/english)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_english
document(path\=wizard/en-nat)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_en-nat
document(path\=wizard/en-other)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_en-other
document(path\=wizard/en-study-work)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_en-study-work
document(path\=wizard/en-test)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_en-test
document(path\=wizard/eu)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_eu
document(path\=wizard/finance)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_finance
document(path\=wizard/german)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_german
document(path\=wizard/gsv$)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_gsv
document(path\=wizard/gsv-document)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_gsv-document
document(path\=wizard/special-treatment$)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_special-t\
    reatment
document(path\=wizard/special-treatment-document)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_s\
    pecial-treatment-document
document(path\=wizard/opt-work-exp$)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_opt-work-exp
document(path\=wizard/opt-work-exp_details)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_opt-wor\
    k-exp_details
document(path\=wizard/personal)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_personal
document(path\=wizard/research-focus)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_research-focus
document(path\=wizard/research-focus)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div(htmlId\=content_wizard_research\
    -focus)/fieldset[0]/div=CLEAR_INDEX
document(path\=wizard/university)/html/body/div[0]/div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[1]=content_wizard_university

body/p/small/a=imprint-link

body/div[0]/ul=breadcrumbs
document(path\=accounts)/html/body/div[0]/ul/li[1]/a=breadcrumb_login-link
document(path\=wizard)/html/body/div[0]/ul/li[1]/a=breadcrumb_logout-link
document(path\=wizard)/html/body/div[0]/ul/li[2]/a=breadcrumb_change-password-link
document(path\=application)/html/body/div[0]/ul/li[1]/a=breadcrumb_logout-link
document(path\=application)/html/body/div[0]/ul/li[2]/a=breadcrumb_change-password-link
document(path\=review)/html/body/div[0]/ul/li[1]/a=breadcrumb_logout-link
document(path\=review)/html/body/div[0]/ul/li[2]/a=breadcrumb_change-password-link

document(path\=accounts/login)/html/body/div[0]/div[1]/div[0]/form/p/a=password-reset-link
document(path\=accounts/login)/html/body/div[0]/div[1]/div[0]/form/div/button[0]=login-btn
document(path\=accounts/login)/html/body/div[0]/div[1]/div[0]/form/div/button[1]=create-account-btn
document(path\=accounts/login)/html/body/div[0]/div[1]/div[0]/form/div/a[0]=create-account-link

document(path\=accounts/password_change)/html/body/div[0]/div[1]/div[0]/form/div[4]/button[0]=change-password-btn

document(path\=accounts/password_reset$)/html/body/div[0]/div[1]/div[0]/form/div[2]/button[0]=reset-password-btn

#################### application side

div(htmlId\=content_application_edit)/form[0]/div[2]/div[0]/button[0]=confirm-btn
div(htmlId\=content_application_edit)/form[0]/div[2]/div[0]/a[0]=back-btn

div(htmlId\=content_application_withdraw)/form[0]/div[2]/div[0]/button[0]=confirm-btn
div(htmlId\=content_application_withdraw)/form[0]/div[2]/div[0]/a[0]=back-btn

div(htmlId\=content_wizard)/div[0]/form/div[2]/div[0]/button[0]=reset-btn
div(htmlId\=content_wizard)/div[0]/form/div[2]/div[0]/button[1]=next-btn
div(htmlId\=content_wizard)/div[0]/form/div[2]/div[0]/a[0]=first-step-btn
div(htmlId\=content_wizard)/div[0]/form/div[2]/div[0]/a[1]=previous-step-btn

div(htmlId\=div_id_university)/div[0]=university-selector
div(htmlId\=div_id_degree_type)/div[0]=degree-type-selector

div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[0]=wizard-steps-headline
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[1]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[2]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[3]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[4]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[5]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[6]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[7]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[8]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[9]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[10]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[11]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[12]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[13]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[14]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[15]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[16]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[17]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[18]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[19]=wizard-steps-navigation
div(htmlId\=content_wizard)/div[0]/form[0]/div[1]/div[0]/ul/li[20]=wizard-steps-navigation

li(htmlId\=wizard-steps-navigation)/a=wizard-steps-navigation-link

document(path\=wizard$)/html/body/div[1]=CLEAR_INDEX,date-chooser-wizard
document(path\=wizard$)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=wizard$)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=wizard$)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=wizard$)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=wizard$)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=wizard/additional-degree_details)/html/body/div[1]=CLEAR_INDEX,date-chooser
document(path\=wizard/additional-degree_details)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=wizard/additional-degree_details)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=wizard/additional-degree_details)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=wizard/additional-degree_details)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=wizard/additional-degree_details)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=wizard/degree)/html/body/div[1]=CLEAR_INDEX,date-chooser-wizard-degree
document(path\=wizard/degree)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=wizard/degree)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=wizard/degree)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=wizard/degree)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=wizard/degree)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=wizard/en-test)/html/body/div[2]=CLEAR_INDEX,date-chooser

document(path\=wizard/opt-work-exp_details)/html/body/div[1]=CLEAR_INDEX,date-chooser-wizard-opt-work-exp_details
document(path\=wizard/opt-work-exp_details)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=wizard/opt-work-exp_details)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=wizard/opt-work-exp_details)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=wizard/opt-work-exp_details)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=wizard/opt-work-exp_details)/html/body/div[6]=CLEAR_INDEX,date-chooser
document(path\=wizard/opt-work-exp_details)/html/body/div[7]=CLEAR_INDEX,date-chooser

document(path\=wizard/personal)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=wizard/personal)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=wizard/personal)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=wizard/personal)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=wizard/personal)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=application/edit_degree$)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree$)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree$)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree$)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree$)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=application/edit_degree_additional-degree_details)/html/body/div[1]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree_additional-degree_details)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree_additional-degree_details)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree_additional-degree_details)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree_additional-degree_details)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=application/edit_degree_additional-degree_details)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=application/edit_personal)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=application/edit_personal)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=application/edit_personal)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=application/edit_personal)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=application/edit_personal)/html/body/div[6]=CLEAR_INDEX,date-chooser

document(path\=application/edit_work_details)/html/body/div[2]=CLEAR_INDEX,date-chooser
document(path\=application/edit_work_details)/html/body/div[3]=CLEAR_INDEX,date-chooser
document(path\=application/edit_work_details)/html/body/div[4]=CLEAR_INDEX,date-chooser
document(path\=application/edit_work_details)/html/body/div[5]=CLEAR_INDEX,date-chooser
document(path\=application/edit_work_details)/html/body/div[6]=CLEAR_INDEX,date-chooser

div(htmlId\=date-chooser)/div[0]=date-chooser_day
div(htmlId\=date-chooser)/div(htmlId\=date-chooser_day)/table[0]=date-chooser_day-table
div(htmlId\=date-chooser)/div[1]=date-chooser_month
div(htmlId\=date-chooser)/div(htmlId\=date-chooser_month)/table[0]=date-chooser_month-table
div(htmlId\=date-chooser)/div[2]=date-chooser_year
div(htmlId\=date-chooser)/div(htmlId\=date-chooser_year)/table[0]=date-chooser_year-table

table(htmlId\=date-chooser_year-table)/thead[0]/tr[0]/th[0]/i[0]=date-chooser_prev-years
table(htmlId\=date-chooser_year-table)/thead[0]/tr[0]/th[2]/i[0]=date-chooser_next-years


table(htmlId\=overview)/tbody[0]/tr/td/form=

form(htmlId\=index_table)/div/table/tbody[0]/tr=CLEAR_INDEX
table(htmlId\=overview)/tbody[0]/tr=CLEAR_INDEX
table(htmlId\=overview)/tbody[0]/tr/td[0]=app_overview-left
table(htmlId\=overview)/tbody[0]/tr/td[0]/a=app_overview_edit-link
table(htmlId\=overview)/tbody[0]/tr/td[1]=app_overview
table(htmlId\=overview)/tbody[0]/tr/td[2]=app_overview

td(htmlId\=app_overview)/a=app_view-uploaded-file-link
td(htmlId\=app_overview)/form[0]/input_submit(htmlId\=submit-id-submit)=app_upload-file-btn

div(htmlId\=content_application_overview)/form=submit_form
form(htmlId\=submit_form)/div[2]/div[0]=submit_div
form(htmlId\=submit_form)/div[4]=submit_div
form(htmlId\=submit_form)/div[6]=submit_div
div(htmlId\=submit_div)/button[0]=submit-btn


########################## reviewer side

document(path\=review/overview)/html/body/div[0]/div[1]/div[0]/div[0]/div[0]/form(htmlId\=index_table)/div[0]/table[0]/tbody[0]/tr[0]/td[1]/\
    a[0]=rev_overview_applicant-link
document(path\=review/incomp)/html/body/div[0]/div[1]/div[0]/div[0]/div[0]/form(htmlId\=index_table)/div[0]/table[0]/tbody[0]/tr[0]/td[1]/a[\
    0]=rev_incomp_applicant-link
document(path\=same_university)/html/body/div[0]/div[1]/div[0]/div[0]/div[0]/form(htmlId\=index_table)/div[0]/table[0]/tbody[0]/tr[0]/td[1]/\
    a[0]=rev_same-uni_applicant-link

div(htmlId\=personal_data-wrapper)=
div(htmlId\=degree_data-wrapper)=
div(htmlId\=gsv_data-wrapper)=
div(htmlId\=english_native_data-wrapper)=
div(htmlId\=english_test_data-wrapper)=
div(htmlId\=german_none_data-wrapper)=
div(htmlId\=st_data-wrapper)=

div(htmlId\=edit-personal_data)=
div(htmlId\=edit-degree_data)=
div(htmlId\=edit-gsv_data)=
div(htmlId\=edit-english_native_data)=
div(htmlId\=edit-english_test_data)=
div(htmlId\=edit-german_none_data)=
div(htmlId\=edit-st_data)=

table(htmlId\=personal_data)=
table(htmlId\=degree_data)=
table(htmlId\=gsv_data)=
table(htmlId\=english_native_data)=
table(htmlId\=english_test_data)=
table(htmlId\=german_none_data)=
table(htmlId\=st_data)=

tr(htmlId\=profile__photo)/td[1]/a[0]=rev_details_photo-link
tr(htmlId\=profile__cv)/td[1]/a[0]=rev_details_cv-link
tr(htmlId\=degree__certificate_tos)/td[1]/a[0]=rev_details_certificate-tos-link
tr(htmlId\=english__en-test__certificate)/td[1]/a[0]=rev_details_english-test-link

div(htmlId\=personal-profile__photo)/a[0]=rev_details_edit-photo-status-btn
div(htmlId\=review_edit-language-status-details)/a[0]=rev_details_edit-language-status-btn
div(htmlId\=grade-degree)/a[0]=rev_details_edit-degree-status-btn
div(htmlId\=grade-degree__cgpa)/a[0]=rev_details_edit-degree-cgpa-status-btn
div(htmlId\=file-degree__certificate_tos)/a[0]=rev_details_edit-certificate-tos-status-btn
div(htmlId\=gsv-gsv)/a[0]=rev_details_edit-certificate-tos-status-btn
div(htmlId\=none-german__de-none)/a[0]=rev_details_edit-german-status-btn
div(htmlId\=file-english__en-test__certificate)/a[0]=rev_details_edit-english-file-status-btn
div(htmlId\=langtest-english__en-test)/a[0]=rev_details_edit-english-status-btn

div(htmlId\=full-personal-profile__photo)=rev_details_edit-status-details
div(htmlId\=full-none-german__de-none)=rev_details_edit-status-details
div(htmlId\=full-grade-degree)=rev_details_edit-status-details
div(htmlId\=full-grade-degree__cgpa)=rev_details_edit-status-details
div(htmlId\=full-file-degree__certificate_tos)=rev_details_edit-status-details
div(htmlId\=full-gsv-gsv)=rev_details_edit-status-details
div(htmlId\=full-file-english__en-test__certificate)=rev_details_edit-status-details
div(htmlId\=full-langtest-english__en-test)=rev_details_edit-status-details

div(htmlId\=rev_details_edit-status-details)/form[0]/div=
div(htmlId\=rev_details_edit-status-details)/form[0]/div/div[0]=rev_details_status-selector-wrapper
div(htmlId\=rev_details_edit-status-details)/form[0]/div/div[0]/select=rev_details_status-selector
div(htmlId\=rev_details_edit-status-details)/form[0]/div/div[0]/textarea=rev_details_status-comment
div(htmlId\=rev_details_edit-status-details)/form[0]/input_button(htmlId\=button-id-submit)=rev_details_submit-status_btn


table(htmlId\=review_data)/tbody[0]/tr[1]/td[0]/button[0]=rev_details_edit-overall-rev-btn
table(htmlId\=review_data)/tbody[0]/tr[1]/td[2]/div[0]/div/div[0]/select(htmlId\=id_status)=rev_details_overall-status-selector
table(htmlId\=review_data)/tbody[0]/tr[2]/td[1]/div[0]/div/div[0]/select(htmlId\=id_reviewer)=rev_details_reviewer-selector



table(htmlId\=furtherdegree_data)/tbody[0]/tr(htmlId\=block__furtherdegree__#__certificate_tos)=block__furtherdegree__X__certificate_tos



div(htmlId\=div_id_base_workmembership-letter-#)=div_id_base_workmembership-letter-X
div(htmlId\=div_id_base_workmembership-letter-##)=div_id_base_workmembership-letter-X
div(htmlId\=div_id_base_workmembership-letter-###)=div_id_base_workmembership-letter-X

input_file(htmlId\=id_base_workmembership-letter-#)=id_base_workmembership-letter-X
input_file(htmlId\=id_base_workmembership-letter-##)=id_base_workmembership-letter-X
input_file(htmlId\=id_base_workmembership-letter-###)=id_base_workmembership-letter-X

input_checkbox(htmlId\=base_workmembership-letter-#-clear_id)=base_workmembership-letter-X-clear_id
input_checkbox(htmlId\=base_workmembership-letter-##-clear_id)=base_workmembership-letter-X-clear_id
input_checkbox(htmlId\=base_workmembership-letter-###-clear_id)=base_workmembership-letter-X-clear_id



div(htmlId\=div_id_base_studymembership-letter-#)=div_id_base_studymembership-letter-X
div(htmlId\=div_id_base_studymembership-letter-##)=div_id_base_studymembership-letter-X
div(htmlId\=div_id_base_studymembership-letter-###)=div_id_base_studymembership-letter-X

input_file(htmlId\=id_base_studymembership-letter-#)=id_base_studymembership-letter-X
input_file(htmlId\=id_base_studymembership-letter-##)=id_base_studymembership-letter-X
input_file(htmlId\=id_base_studymembership-letter-###)=id_base_studymembership-letter-X



div(htmlId\=div_id_base_bachelormembership-letter-#)=div_id_base_bachelormembership-letter-X
div(htmlId\=div_id_base_bachelormembership-letter-##)=div_id_base_bachelormembership-letter-X
div(htmlId\=div_id_base_bachelormembership-letter-###)=div_id_base_bachelormembership-letter-X

input_file(htmlId\=id_base_bachelormembership-letter-#)=id_base_bachelormembership-letter-X
input_file(htmlId\=id_base_bachelormembership-letter-##)=id_base_bachelormembership-letter-X
input_file(htmlId\=id_base_bachelormembership-letter-###)=id_base_bachelormembership-letter-X

input_checkbox(htmlId\=base_bachelormembership-letter-#-clear_id)=base_bachelormembership-letter-X-clear_id
input_checkbox(htmlId\=base_bachelormembership-letter-##-clear_id)=base_bachelormembership-letter-X-clear_id
input_checkbox(htmlId\=base_bachelormembership-letter-###-clear_id)=base_bachelormembership-letter-X-clear_id



div(htmlId\=div_id_base_bachelormembership_#-letter)=div_id_base_bachelormembership_X-letter
div(htmlId\=div_id_base_bachelormembership_##-letter)=div_id_base_bachelormembership_X-letter
div(htmlId\=div_id_base_bachelormembership_###-letter)=div_id_base_bachelormembership_X-letter

input_file(htmlId\=id_base_bachelormembership_#-letter)=id_base_bachelormembership_X-letter
input_file(htmlId\=id_base_bachelormembership_##-letter)=id_base_bachelormembership_X-letter
input_file(htmlId\=id_base_bachelormembership_###-letter)=id_base_bachelormembership_X-letter

p(htmlId\=hint_id_base_bachelormembership_#-letter)=hint_id_base_bachelormembership_X-letter
p(htmlId\=hint_id_base_bachelormembership_##-letter)=hint_id_base_bachelormembership_X-letter
p(htmlId\=hint_id_base_bachelormembership_###-letter)=hint_id_base_bachelormembership_X-letter



div(htmlId\=div_id_base_studymembership_#-letter)=div_id_base_studymembership_X-letter
div(htmlId\=div_id_base_studymembership_##-letter)=div_id_base_studymembership_X-letter
div(htmlId\=div_id_base_studymembership_###-letter)=div_id_base_studymembership_X-letter

input_file(htmlId\=id_base_studymembership_#-letter)=id_base_studymembership_X-letter
input_file(htmlId\=id_base_studymembership_##-letter)=id_base_studymembership_X-letter
input_file(htmlId\=id_base_studymembership_###-letter)=id_base_studymembership_X-letter



div(htmlId\=div_id_base_workmembership_#-letter)=div_id_base_workmembership_X-letter
div(htmlId\=div_id_base_workmembership_##-letter)=div_id_base_workmembership_X-letter
div(htmlId\=div_id_base_workmembership_###-letter)=div_id_base_workmembership_X-letter

input_file(htmlId\=id_base_workmembership_#-letter)=id_base_workmembership_X-letter
input_file(htmlId\=id_base_workmembership_##-letter)=id_base_workmembership_X-letter
input_file(htmlId\=id_base_workmembership_###-letter)=id_base_workmembership_X-letter

p(htmlId\=hint_id_base_workmembership_#-letter)=hint_id_base_workmembership_X-letter
p(htmlId\=hint_id_base_workmembership_##-letter)=hint_id_base_workmembership_X-letter
p(htmlId\=hint_id_base_workmembership_###-letter)=hint_id_base_workmembership_X-letter



div(htmlId\=div_id_base_languagetest_#-certificate)=div_id_base_languagetest_X-certificate
div(htmlId\=div_id_base_languagetest_##-certificate)=div_id_base_languagetest_X-certificate
div(htmlId\=div_id_base_languagetest_###-certificate)=div_id_base_languagetest_X-certificate

input_file(htmlId\=id_base_languagetest_#-certificate)=id_base_languagetest_X-certificate
input_file(htmlId\=id_base_languagetest_##-certificate)=id_base_languagetest_X-certificate
input_file(htmlId\=id_base_languagetest_###-certificate)=id_base_languagetest_X-certificate

p(htmlId\=hint_id_base_languagetest_#-certificate)=hint_id_base_languagetest_X-certificate
p(htmlId\=hint_id_base_languagetest_##-certificate)=hint_id_base_languagetest_X-certificate
p(htmlId\=hint_id_base_languagetest_###-certificate)=hint_id_base_languagetest_X-certificate



div(htmlId\=div_id_base_workingexperience_#-reference_letter)=div_id_base_workingexperience_X-reference_letter
div(htmlId\=div_id_base_workingexperience_##-reference_letter)=div_id_base_workingexperience_X-reference_letter
div(htmlId\=div_id_base_workingexperience_###-reference_letter)=div_id_base_workingexperience_X-reference_letter

input_file(htmlId\=id_base_workingexperience_#-reference_letter)=id_base_workingexperience_X-reference_letter
input_file(htmlId\=id_base_workingexperience_##-reference_letter)=id_base_workingexperience_X-reference_letter
input_file(htmlId\=id_base_workingexperience_###-reference_letter)=id_base_workingexperience_X-reference_letter

p(htmlId\=hint_id_base_workingexperience_#-reference_letter)=hint_id_base_workingexperience_X-reference_letter
p(htmlId\=hint_id_base_workingexperience_##-reference_letter)=hint_id_base_workingexperience_X-reference_letter
p(htmlId\=hint_id_base_workingexperience_###-reference_letter)=hint_id_base_workingexperience_X-reference_letter



div(htmlId\=div_id_base_applicantprofile_#-photo)=div_id_base_applicantprofile_X-photo
div(htmlId\=div_id_base_applicantprofile_##-photo)=div_id_base_applicantprofile_X-photo
div(htmlId\=div_id_base_applicantprofile_###-photo)=div_id_base_applicantprofile_X-photo

input_file(htmlId\=id_base_applicantprofile_#-photo)=id_base_applicantprofile_X-photo
input_file(htmlId\=id_base_applicantprofile_##-photo)=id_base_applicantprofile_X-photo
input_file(htmlId\=id_base_applicantprofile_###-photo)=id_base_applicantprofile_X-photo

p(htmlId\=hint_id_base_applicantprofile_#-photo)=hint_id_base_applicantprofile_X-photo
p(htmlId\=hint_id_base_applicantprofile_##-photo)=hint_id_base_applicantprofile_X-photo
p(htmlId\=hint_id_base_applicantprofile_###-photo)=hint_id_base_applicantprofile_X-photo



div(htmlId\=div_id_base_applicantprofile_#-cv)=div_id_base_applicantprofile_X-cv
div(htmlId\=div_id_base_applicantprofile_##-cv)=div_id_base_applicantprofile_X-cv
div(htmlId\=div_id_base_applicantprofile_###-cv)=div_id_base_applicantprofile_X-cv

input_file(htmlId\=id_base_applicantprofile_#-cv)=id_base_applicantprofile_X-cv
input_file(htmlId\=id_base_applicantprofile_##-cv)=id_base_applicantprofile_X-cv
input_file(htmlId\=id_base_applicantprofile_###-cv)=id_base_applicantprofile_X-cv

p(htmlId\=hint_id_base_applicantprofile_#-cv)=hint_id_base_applicantprofile_X-cv
p(htmlId\=hint_id_base_applicantprofile_##-cv)=hint_id_base_applicantprofile_X-cv
p(htmlId\=hint_id_base_applicantprofile_###-cv)=hint_id_base_applicantprofile_X-cv



div(htmlId\=div_id_base_degree_#-certificate_tos)=div_id_base_degree_X-certificate_tos
div(htmlId\=div_id_base_degree_##-certificate_tos)=div_id_base_degree_X-certificate_tos
div(htmlId\=div_id_base_degree_###-certificate_tos)=div_id_base_degree_X-certificate_tos

input_file(htmlId\=id_base_degree_#-certificate_tos)=id_base_degree_X-certificate_tos
input_file(htmlId\=id_base_degree_##-certificate_tos)=id_base_degree_X-certificate_tos
input_file(htmlId\=id_base_degree_###-certificate_tos)=id_base_degree_X-certificate_tos

p(htmlId\=hint_id_base_degree_#-certificate_tos)=hint_id_base_degree_X-certificate_tos
p(htmlId\=hint_id_base_degree_##-certificate_tos)=hint_id_base_degree_X-certificate_tos
p(htmlId\=hint_id_base_degree_###-certificate_tos)=hint_id_base_degree_X-certificate_tos



div(htmlId\=div_id_base_furtherdegree_#-certificate_tos)=div_id_base_furtherdegree_X-certificate_tos
div(htmlId\=div_id_base_furtherdegree_##-certificate_tos)=div_id_base_furtherdegree_X-certificate_tos
div(htmlId\=div_id_base_furtherdegree_###-certificate_tos)=div_id_base_furtherdegree_X-certificate_tos

input_file(htmlId\=id_base_furtherdegree_#-certificate_tos)=id_base_furtherdegree_X-certificate_tos
input_file(htmlId\=id_base_furtherdegree_##-certificate_tos)=id_base_furtherdegree_X-certificate_tos
input_file(htmlId\=id_base_furtherdegree_###-certificate_tos)=id_base_furtherdegree_X-certificate_tos



div(htmlId\=div_id_compscience_gradingsystemvalidationdocument_#-document)=div_id_compscience_gradingsystemvalidationdocument_X-document
div(htmlId\=div_id_compscience_gradingsystemvalidationdocument_##-document)=div_id_compscience_gradingsystemvalidationdocument_X-document
div(htmlId\=div_id_compscience_gradingsystemvalidationdocument_###-document)=div_id_compscience_gradingsystemvalidationdocument_X-document

input_file(htmlId\=id_compscience_gradingsystemvalidationdocument_#-document)=id_compscience_gradingsystemvalidationdocument_X-document
input_file(htmlId\=id_compscience_gradingsystemvalidationdocument_##-document)=id_compscience_gradingsystemvalidationdocument_X-document
input_file(htmlId\=id_compscience_gradingsystemvalidationdocument_###-document)=id_compscience_gradingsystemvalidationdocument_X-document

p(htmlId\=hint_id_compscience_gradingsystemvalidationdocument_#-document)=hint_id_compscience_gradingsystemvalidationdocument_X-document
```

```
p(htmlId\=hint_id_compscience_gradingsystemvalidationdocument_##-document)=hint_id_compscience_gradingsystemvalidationdocument_X-document
p(htmlId\=hint_id_compscience_gradingsystemvalidationdocument_###-document)=hint_id_compscience_gradingsystemvalidationdocument_X-document



div(htmlId\=div_id_base_otherlanguageproof_#-document)=div_id_base_otherlanguageproof_X-document
div(htmlId\=div_id_base_otherlanguageproof_##-document)=div_id_base_otherlanguageproof_X-document
div(htmlId\=div_id_base_otherlanguageproof_###-document)=div_id_base_otherlanguageproof_X-document

input_file(htmlId\=id_base_otherlanguageproof_#-document)=id_base_otherlanguageproof_X-document
input_file(htmlId\=id_base_otherlanguageproof_##-document)=id_base_otherlanguageproof_X-document
input_file(htmlId\=id_base_otherlanguageproof_###-document)=id_base_otherlanguageproof_X-document

p(htmlId\=hint_id_base_otherlanguageproof_#-document)=hint_id_base_otherlanguageproof_X-document
p(htmlId\=hint_id_base_otherlanguageproof_##-document)=hint_id_base_otherlanguageproof_X-document
p(htmlId\=hint_id_base_otherlanguageproof_###-document)=hint_id_base_otherlanguageproof_X-document



div(htmlId\=div_id_base_nativespeaker_#-document)=div_id_base_nativespeaker_X-document
div(htmlId\=div_id_base_nativespeaker_##-document)=div_id_base_nativespeaker_X-document
div(htmlId\=div_id_base_nativespeaker_###-document)=div_id_base_nativespeaker_X-document

input_file(htmlId\=id_base_nativespeaker_#-document)=id_base_nativespeaker_X-document
input_file(htmlId\=id_base_nativespeaker_##-document)=id_base_nativespeaker_X-document
input_file(htmlId\=id_base_nativespeaker_###-document)=id_base_nativespeaker_X-document

p(htmlId\=hint_id_base_nativespeaker_#-document)=hint_id_base_nativespeaker_X-document
p(htmlId\=hint_id_base_nativespeaker_##-document)=hint_id_base_nativespeaker_X-document
p(htmlId\=hint_id_base_nativespeaker_###-document)=hint_id_base_nativespeaker_X-document
```

B.2. Parsing Configuration for Case Study 2

For the second case study, two parsing configuration files were created, one for the old, and one for the new version of the website. This was required, as the website does not make much use of DOM element ids, and because the old and the new version were too similar to be able to configure the parsing with only one configuration file. The configuration files had the following content:

B.2.1. Parsing Configuration for the Old Website Version

```
##################### Mapping of document paths
document(path\=/awards$)=awards/overview
document(path\=/awards/index.php$)=awards/overview
document(path\=/awards/###############)=awards/details
document(path\=/certified-tester)=other/certified-tester
document(path\=/emc-academic-alliance)=other/emc-academic-alliance
document(path\=/external-academic-services/######)=staff/external-academic-services/details
document(path\=/external-academic-services-taxonomy/######)=admin/staff/external-academic-services/taxonomy
document(path\=/filter/tips)=admin/help/filter/tips
document(path\=/former-staff)=staff/former/overview
document(path\=/help/facetapi_bonus/README.txt)=admin/help/facetapi_bonus
document(path\=/how-to-find-us)=home/how-to-find-us
document(path\=/import)=admin/import
document(path\=/imprint)=other/imprint
document(path\=^/index.php$)=/
document(path\=/ingo-tributh)=CLEAR_QUERY,staff/details
document(path\=/internal-academic-services/######)=staff/internal-academic-services/details
document(path\=/jobs$)=staff/jobs/overview
document(path\=/jobs/index.php$)=staff/jobs/overview
document(path\=/jobss$)=staff/jobs/overview
document(path\=/jobss/$)=staff/jobs/overview
document(path\=/jobss/######)=staff/jobs/details
document(path\=/lectures$)=lectures/overview
document(path\=/lectures/$)=lectures/overview
document(path\=/lectures/######)=CLEAR_QUERY,lectures/details
document(path\=/news$)=CLEAR_QUERY,home/news/overview
document(path\=/news/$)=CLEAR_QUERY,home/news/overview
document(path\=/news/######)=CLEAR_QUERY,home/news/details
document(path\=/news/index.php$)=CLEAR_QUERY,home/news/overview
document(path\=/node/$)=admin/nodes/recently-added
document(path\=/node/####$)=publications/details
document(path\=/our-research$)=research/overview
document(path\=/publications$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/######)=CLEAR_QUERY,publications/details
document(path\=/publications-recent$)=CLEAR_QUERY,publications/recent/overview

document(path\=/publications/A-98-04/Report-A-98-04.ps.gz$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/BZ_DV_IS_HN_JG/main.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/BZ_HN_JG_DE_PB/trex.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/EE/Masterarbeit_SRE_Eduard_Enriquez_28072008-1.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ER_IS_JG/SAM2000RudolphEtAll.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ER_JG_PG/SDL99-Harmonization.ps.gz$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/EW_JG_ST_BZ/WSTestFramework_main.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/FBT98/FBT98.ps.gz$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/HN_BZ_JG_PB_DE/stvr_quality_assurance_for_ttcn3_test_specifications.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/HN_BZ/wrt_ecoop2007.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/HN_MB/ttcn3codesmells_testcom2007.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/IS_ZD_JG_AR/TestCom2003_UTP_Final.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/JG_ER/MSC-Survey93.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/JG/Grabowski.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/JK/bmsc_thesis_kemnade.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/MB/bisanz_mastersthesis.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/PB_ZD_JG_ME_GK_IS_PG/U2TP-CONQUEST-2004.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ST/WebServiceTestFrameworkWithTTCN-3.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/TR_AA_JM_FV/rings-gb74.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/TR_HN_JG/rings-testingGridApplication.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ZD_JG_HN_HP/jzus.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ZD_JG_HN_HP/testcom2004_from_design_to_test_with_uml.pdf$)=CLEAR_QUERY,publications/all/overview

document(path\=/research$)=CLEAR_QUERY,research/overview
document(path\=/research/######)=research/project-details
document(path\=/search/node/)=search-results
document(path\=/semester/ws2014)=admin/taxonomy/list

document(path\=^/staff$)=CLEAR_QUERY,staff/current/overview
document(path\=^/staff/$)=CLEAR_QUERY,staff/current/overview
document(path\=^/staff/index.php$)=CLEAR_QUERY,staff/current/overview
document(path\=^/staff/#######)=CLEAR_QUERY,staff/details
document(path\=^/staff/single_staff/index.php)=CLEAR_QUERY,staff/current/overview

document(path\=/start$)=/
document(path\=/taxonomy/term/#)=admin/taxonomy/term/details
document(path\=/teaching$)=CLEAR_QUERY,teaching/overview
document(path\=/teaching-all$)=CLEAR_QUERY,teaching/overview
document(path\=/teaching/writing-and-presenting)=teaching/writing-and-presenting
document(path\=/user$)=CLEAR_QUERY,user/login
document(path\=/user/login$)=CLEAR_QUERY,user/login
document(path\=/user/logout$)=CLEAR_QUERY,user/logout
document(path\=/user/password$)=CLEAR_QUERY,user/password
document(path\=/user/reset/###/###################################################)=user/password-reset
document(path\=/users/$)=admin/users/overview
document(path\=/users/admin$)=admin/users/overview
document(path\=/users/sadhatarao$)=admin/users/details
document(path\=/users/students$)=admin/users/details
document(path\=/users/vhonsel$)=admin/users/details

document(path\=/we-wish-you-all-merry-christmas-and-happy-new-year$)=/
document(path\=/we-wish-you-merry-christmas-and-happy-new-year$)=/

document(path\=/compscience/accounts/$)=CLEAR_QUERY,invalid-page
document(path\=/compscience/accounts/login/)=CLEAR_QUERY,invalid-page
document(path\=/compscience/application/)=CLEAR_QUERY,invalid-page
document(path\=/edu$)=CLEAR_QUERY,invalid-page
document(path\=/edu/$)=CLEAR_QUERY,invalid-page
document(path\=/edu/eval.php)=CLEAR_QUERY,invalid-page
document(path\=/edu/lv.php)=CLEAR_QUERY,invalid-page
document(path\=/edu/notes/index.php)=CLEAR_QUERY,invalid-page
document(path\=/favicon.ico)=CLEAR_QUERY,invalid-page
document(path\=/intranet/bibo/index.php)=CLEAR_QUERY,invalid-page
document(path\=/itis/2015/suse/accounts/login/)=CLEAR_QUERY,invalid-page
document(path\=/itis/2015/suse/application/)=CLEAR_QUERY,invalid-page
document(path\=/itis/accounts/login/)=CLEAR_QUERY,invalid-page
document(path\=/itis/accounts/password/change/)=CLEAR_QUERY,invalid-page
document(path\=/itis/accounts/password/reset/)=CLEAR_QUERY,invalid-page
document(path\=/itis/application/)=CLEAR_QUERY,invalid-page
document(path\=/itis/imprint/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/$)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/41/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/500/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/78/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/login/)=CLEAR_QUERY,invalid-page
document(path\=/itis/wizard/)=CLEAR_QUERY,invalid-page
document(path\=/lecture$)=CLEAR_QUERY,invalid-page
document(path\=/lectures/exam-ss2014)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ss2014)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ws2003)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ws2013)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ws2014)=CLEAR_QUERY,invalid-page
document(path\=/notes/$)=CLEAR_QUERY,invalid-page
document(path\=/notes/SS2004/Grabowski/I-Einfuehrung-6-auf-1.pdf)=CLEAR_QUERY,invalid-page
document(path\=/notes/SS2007/grabowski/0-Allgemeines-Organisatorisches-6-auf-1.pdf)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2004/$)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2004/neukirchen/$)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2004/neukirchen/case-tools_Noedler.pdf)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2008/bzeiss/51/07collectiveownership.pdf)=CLEAR_QUERY,invalid-page
document(path\=/pubs/pub_liste/pub_anzeige.php)=CLEAR_QUERY,invalid-page
document(path\=/pubs/single_pub/index.php)=CLEAR_QUERY,invalid-page
document(path\=/research/main_projects/projects/single_projects/index.php)=CLEAR_QUERY,invalid-page
document(path\=/se$)=CLEAR_QUERY,invalid-page
document(path\=/sites/all/themes/samara/styles/custom.css)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/pages/)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/DissertationGrabowski.pdf)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/main1.pdf)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/PhD_wafi.pdf)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/pictures/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/pictures/IMG_0466.JPG%3Fitok%3DJSPAGeSr)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/pictures/verena10_0.JPG%3Fitok%3Dgm4kHYS0)=CLEAR_QUERY,invalid-page
document(path\=/theses/criteria.php)=CLEAR_QUERY,invalid-page
document(path\=/theses/index.php)=CLEAR_QUERY,invalid-page
document(path\=/trac$)=CLEAR_QUERY,invalid-page
document(path\=/trac/wiki/TracIni)=CLEAR_QUERY,invalid-page

#################### Mapping major page contents
document(path\=/$)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=main)/div(htm\
        lId\=page)=page_root
document(path\=admin/help/facetapi_bonus)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/di\
        v(htmlId\=main)/div(htmlId\=page)=page_admin_help_facetapi_bonus
document(path\=admin/help/filter/tips)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(h\
        tmlId\=main)/div(htmlId\=page)=page_admin_help_filter_tips
document(path\=admin/import)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=mai\
        n)/div(htmlId\=page)=page_admin_import
document(path\=admin/nodes/recently-added)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/d\
        iv(htmlId\=main)/div(htmlId\=page)=page_admin_nodes_recently-added
document(path\=admin/staff/external-academic-services/taxonomy)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(ht\
        mlId\=main-wrapper)/div(htmlId\=main)/div(htmlId\=page)=page_admin_staff_external-academic-services_taxonomy
document(path\=admin/taxonomy/list)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
        Id\=main)/div(htmlId\=page)=page_admin_taxonomy_list
document(path\=admin/taxonomy/term/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/\
        div(htmlId\=main)/div(htmlId\=page)=page_admin_taxonomy_term_details
document(path\=admin/users/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
        Id\=main)/div(htmlId\=page)=page_admin_users_details
document(path\=admin/users/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htm\
        lId\=main)/div(htmlId\=page)=page_admin_users_overview
document(path\=awards/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=m\
        ain)/div(htmlId\=page)=page_awards_details
document(path\=awards/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=\
        main)/div(htmlId\=page)=page_awards_overview
document(path\=home/how-to-find-us)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
        Id\=main)/div(htmlId\=page)=page_home_how-to-find-us
document(path\=home/news/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
        \=main)/div(htmlId\=page)=page_home_news_details
document(path\=home/news/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlI\
        d\=main)/div(htmlId\=page)=page_home_news_overview
document(path\=lectures/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\\
        =main)/div(htmlId\=page)=page_lectures_details
document(path\=lectures/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_lectures_overview
document(path\=other/certified-tester)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(h\
    tmlId\=main)/div(htmlId\=page)=page_other_certified-tester
document(path\=other/emc-academic-alliance)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/\
    div(htmlId\=main)/div(htmlId\=page)=page_other_emc-academic-alliance
document(path\=other/imprint)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=ma\
    in)/div(htmlId\=page)=page_other_imprint
document(path\=publications/all/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/di\
    v(htmlId\=main)/div(htmlId\=page)=page_publications_all_overview
document(path\=publications/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htm\
    lId\=main)/div(htmlId\=page)=page_publications_details
document(path\=publications/recent/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)\
    /div(htmlId\=main)/div(htmlId\=page)=page_publications_recent_overview
document(path\=research/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_research_overview
document(path\=research/project-details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div\
    (htmlId\=main)/div(htmlId\=page)=page_research_project-details
document(path\=search-results)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=m\
    ain)/div(htmlId\=page)=page_search-results
document(path\=staff/current/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(h\
    tmlId\=main)/div(htmlId\=page)=page_staff_current_overview
document(path\=staff/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=ma\
    in)/div(htmlId\=page)=page_staff_details
document(path\=staff/external-academic-services/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=m\
    ain-wrapper)/div(htmlId\=main)/div(htmlId\=page)=page_staff_external-academic-services_details
document(path\=staff/former/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(ht\
    mlId\=main)/div(htmlId\=page)=page_staff_former_overview
document(path\=staff/internal-academic-services/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=m\
    ain-wrapper)/div(htmlId\=main)/div(htmlId\=page)=page_staff_internal-academic-services_details
document(path\=staff/jobs/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlI\
    d\=main)/div(htmlId\=page)=page_staff_jobs_details
document(path\=staff/jobs/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_staff_jobs_overview
document(path\=teaching/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_teaching_overview
document(path\=teaching/writing-and-presenting)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapp\
    er)/div(htmlId\=main)/div(htmlId\=page)=page_teaching_writing-and-presenting
document(path\=user/login)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=main)\
    /div(htmlId\=page)=page_user_login
document(path\=user/logout)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=main\
    )/div(htmlId\=page)=page_user_logout
document(path\=user/password)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=ma\
    in)/div(htmlId\=page)=page_user_password
document(path\=user/password-reset)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_user_password-reset

#################### Overall
body
div(htmlId\=logo)/a(htmlId\=logo_home)/img=logo_img
body[0]/div(htmlId\=skip-link)/a=skip_link

#################### Start page
div(htmlId\=page)/div/ul[0]/li[0]/a[0]=admin_view
div(htmlId\=page)/div/ul[0]/li[1]/a[0]=admin_edit
div(htmlId\=page)/div/ul[0]/li[2]/a[0]=admin_log

div(htmlId\=node-startpage)/div[0]/div[0]/div[0]/div[0]/div[0]/a[0]=start_staff_link
div(htmlId\=node-startpage)/div[0]/div[0]/div[0]/div[0]/div[1]/a[0]=start_mail

div(htmlId\=node-startpage)/div[0]/div[1]/div[0]/div=CLEAR_INDEX
div(htmlId\=node-startpage)/div[0]/div[1]/div[0]/div/div[0]/span[0]/a[0]=start_newslink

#################### Menu
div(htmlId\=navigation)/ul[0]/li[0]/a=menu_home
div(htmlId\=navigation)/ul[0]/li[0]/ul[0]/li[0]/a=menu_how-to-find-us

div(htmlId\=navigation)/ul[0]/li[1]/a=menu_staff
div(htmlId\=navigation)/ul[0]/li[1]/ul[0]/li[0]/a=menu_staff_current
div(htmlId\=navigation)/ul[0]/li[1]/ul[0]/li[1]/a=menu_staff_jobs
div(htmlId\=navigation)/ul[0]/li[1]/ul[0]/li[2]/a=menu_staff_former

div(htmlId\=navigation)/ul[0]/li[2]/a=menu_research

div(htmlId\=navigation)/ul[0]/li[3]/a=menu_publications
div(htmlId\=navigation)/ul[0]/li[3]/ul[0]/li[0]/a=menu_publications_all
div(htmlId\=navigation)/ul[0]/li[3]/ul[0]/li[1]/a=menu_publications_recent

div(htmlId\=navigation)/ul[0]/li[4]/a=menu_awards
div(htmlId\=navigation)/ul[0]/li[5]/a=menu_teaching
div(htmlId\=navigation)/ul[0]/li[5]/ul[0]/li[0]/a=menu_teaching_lectures
div(htmlId\=navigation)/ul[0]/li[5]/ul[0]/li[1]/a=menu_teaching_writing-and-presenting

#################### Toolbar
ul(htmlId\=toolbar-home)/li[0]/a[0]=toolbar_home
ul(htmlId\=toolbar-user)/li[0]/a[0]=toolbar_userlink
ul(htmlId\=toolbar-user)/li[1]/a[0]=toolbar_logout
div(htmlId\=toolbar)/div[0]/a[0]=hidden_link
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[0]/a[0]=toolbar_add-content
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[1]/a[0]=toolbar_find-content
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[2]/a[0]=toolbar_install-module
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[3]/a[0]=toolbar_taxonomy
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[4]/a[0]=toolbar_feeds
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[5]/a[0]=toolbar_content-types


#################### Other common stuff
div(htmlId\=closure)/div(htmlId\=info)/ul(htmlId\=navlist2)/li[0]/a[0]=closure_imprint
div(htmlId\=closure)/div(htmlId\=info)/ul(htmlId\=navlist2)/li[1]/a[0]=closure_login

div(htmlId\=branding)/div(htmlId\=logo)/a=logo_home
div(htmlId\=branding)/div(htmlId\=name-and-slogan)/div(htmlId\=site-name)/a=slogan_home
div(htmlId\=branding)/div(htmlId\=name-and-slogan)/div(htmlId\=site-slogan)/a[0]=link_ifi
div(htmlId\=branding)/div(htmlId\=name-and-slogan)/div(htmlId\=site-slogan)/a[1]=link_unigoe
div(htmlId\=logo-title)/a=logo_home
div(htmlId\=name-and-slogan)/h1(htmlId\=site-name)/a=slogan_home

#################### Sidebars
div(htmlId\=block-facetapi-dii7dzt2w202fmxdxuz3k6zkhahv17pd)=block-facetapi
div(htmlId\=block-facetapi-ba870gqlmgxg3kakttxrq2y1paet88yj)=block-facetapi
div(htmlId\=block-facetapi-8fztyvn1a9ayhbbsvwslsqun2oz84k0z)=block-facetapi
div(htmlId\=block-facetapi-jsbyxjgad7ybezm3vzrmtaitmhzm21ru)=block-facetapi
div(htmlId\=block-facetapi-08o0hn2r9xl8lxj01sb93yxggnz6cv5i)=block-facetapi
div(htmlId\=block-facetapi)/div=CLEAR_INDEX
div(htmlId\=block-facetapi)/div/div[0]/a[0]=filter_showmore_link
div(htmlId\=block-facetapi)/a=block-facetapi_configure_link
div(htmlId\=block-facetapi)/div/a=block-facetapi_configure_link
div(htmlId\=block-facetapi)/div/ul/li[0]/a=block-facetapi_configure_display
div(htmlId\=block-facetapi)/div/ul/li[1]/a=block-facetapi_configure_dependencies
div(htmlId\=block-facetapi)/div/ul/li[2]/a=block-facetapi_configure_filters
div(htmlId\=block-facetapi)/div/ul/li[3]/a=block-facetapi_configure_block

div(htmlId\=block-current-search-standard)/div=CLEAR_INDEX
div(htmlId\=block-current-search-standard)/div/a=block-current-search_configure_link
div(htmlId\=block-current-search-standard)/div/ul/li[0]/a=block-current-search_configure_items
div(htmlId\=block-current-search-standard)/div/ul/li[1]/a=block-current-search_configure_block

div(htmlId\=block-views-projects-block-2)=block_ongoing_projects
div(htmlId\=block_ongoing_projects)/h2/a=block_project_header
div(htmlId\=block_ongoing_projects)/div[0]=block_project_list
div(htmlId\=block_ongoing_projects)/div[1]=block_project_list
div(htmlId\=block-views-projects-block-1)=block_past_projects
div(htmlId\=block_past_projects)/h2/a=block_project_header
div(htmlId\=block_past_projects)/div[0]=block_project_list
div(htmlId\=block_past_projects)/div[1]=block_project_list

div(htmlId\=block_project_list)/a=block_admin_link
div(htmlId\=block_project_list)/ul[0]/li[0]/a=block_admin_edit-view_link
div(htmlId\=block_project_list)/ul[0]/li[1]/a=block_admin_configure-view_link
div(htmlId\=block_project_list)/div[0]/div[0]/div[0]/ul[0]/li=CLEAR_INDEX
div(htmlId\=block_project_list)/div[0]/div[0]/div[0]/ul[0]/li/div[0]/span[0]/a[0]=project_link

#################### Login page
div(htmlId\=page_user_login)/div=CLEAR_INDEX
div(htmlId\=page_user_login)/div/ul/li[0]/a=login_edit-submit_or_view-link
div(htmlId\=page_user_login)/div/ul/li[1]/a=login_request-new-password_or_edit
div(htmlId\=page_user_login)/div/ul/li[2]/a=login_shortcuts

#################### Root page
div(htmlId\=page_root)/div=CLEAR_INDEX

#################### News details page
div(htmlId\=page_home_news_details)/div=CLEAR_INDEX
div(htmlId\=page_home_news_details)/div/div=node-news-details

#################### News overview page
```

```
div(htmlId\=page_home_news_overview)/div[1]/div[0]/div[0]/div=CLEAR_INDEX
div(htmlId\=page_home_news_details)/div[1]/div[0]/div/div=node-news-list-entry

#################### Lectures details page
div(htmlId\=page_lectures_details)/div=CLEAR_INDEX
div(htmlId\=page_lectures_details)/div/div=node-lectures-details

#################### External Academic Service details page
div(htmlId\=page_staff_external-academic-services_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_external-academic-services_details)/div/div=node-external-academic-services-details
div(htmlId\=page_staff_external-academic-services_details)/div/ul/li=CLEAR_INDEX

#################### External Academic Service details page
div(htmlId\=page_admin_staff_external-academic-services_taxonomy)/div[1]/div=node-external-academic-services-details

#################### Staff details page
div(htmlId\=page_staff_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_details)/div/div=node-staff-details

#################### Publications details page
div(htmlId\=page_publications_details)/div=CLEAR_INDEX
div(htmlId\=page_publications_details)/div/div=node-publication-details

#################### Awards page
div(htmlId\=page_awards_overview)/div[1]/div[0]/div[0]/div[0]/ul[0]/li=award-description

#################### Search results page
div(htmlId\=page_search_results)/div=CLEAR_INDEX
div(htmlId\=page_search_results)/div/ol[0]/li=search-result

#################### Staff list
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr=CLEAR_INDEX
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td=CLEAR_INDEX
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[0]/div[0]/a[0]=stafflist_imglink
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[0]/div[0]/a[0]/img=stafflist_img
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[1]/span[0]/a[0]=stafflist_namelink
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[3]/div[0]/span=stafflist_telno
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[3]/div[0]/span/span=
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[3]/div[0]/span/span/span=
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[4]/div[0]/a[0]=stafflist_email
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[4]/div[0]/p[0]/a[0]=stafflist_email

#################### Jobs details page
div(htmlId\=page_staff_jobs_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_jobs_details)/div/div=node-jobs-details

#################### Jobs overview page
div(htmlId\=page_staff_jobs_overview)/div[1]/div[0]/div[0]/div=CLEAR_INDEX

#################### Teaching list
div(htmlId\=page_teaching_overview)/div[1]=node-teaching-overview
div(htmlId\=page_teaching_overview)/div[2]=node-teaching-overview
div(htmlId\=page_teaching_overview)/div[3]=node-teaching-overview
div(htmlId\=node-teaching-overview)/div[0]/div[0]/div=course-entry
div(htmlId\=course-entry)/div[0]/h4/a=course-link
div(htmlId\=course-entry)/div[1]/span/a=course-staff-link
div(htmlId\=node-teaching-overview)/div[0]/div[1]/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-teaching-overview)/div[0]/div[1]/ul[0]/li/a=pager-link

ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-semester)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-semester)/li/a=teachlist_term_filter
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-staff)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-staff)/li/a=teachlist_staff_filter

#################### Publication list
div(htmlId\=page_publications_all_overview)/div=CLEAR_INDEX
div(htmlId\=page_publications_all_overview)/div/div=node-publication-overview
div(htmlId\=page_publications_all_overview)/div/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-publication-overview)/div[1]/div=publication-type-group
div(htmlId\=node-publication-overview)/div[2]/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-publication-overview)/div[2]/ul[0]/li/a=pager-link

div(htmlId\=publication-type-group)/div=publication-entry
div(htmlId\=publication-type-group)/ul/li=CLEAR_INDEX
div(htmlId\=publication-entry)/ul[0]/li=CLEAR_INDEX
div(htmlId\=publication-entry)/ul[0]/li/div[0]/a=author-link
div(htmlId\=publication-entry)/ul[0]/li/span[0]/em/a=publication-link
div(htmlId\=publication-entry)/ul[0]/li/span[1]/span[0]/a=publication-medium-link
div(htmlId\=page_publications_recent_overview)/div[1]/div[0]/div[0]/div=publication-entry

ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-year)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-year)/li/a=publist_year_filter
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-author)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-author)/li/a=publist_author_filter
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-doc-type)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-doc-type)/li/a=publist_doc-type_filter

#################### Staff details
div(htmlId\=node-staff-details)/div[1]/div[0]/div[0]/div[0]/img=staff_image

div(htmlId\=node-staff-details)/div[2]/div=CLEAR_INDEX
div(htmlId\=node-staff-details)/div[2]/div/div[0]/div[0]/div/div[0]/p[0]/a[0]=staff_email
div(htmlId\=node-staff-details)/div[2]/div/div[0]/div[0]/div/div[0]/a[0]=staff_telno

div(htmlId\=node-staff-details)/div[3]/div[0]/div[0]/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-staff-details)/div[3]/div[0]/div[0]/ul[0]/li/a=staff_verticaltab_selection

fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/a=publist_admin_link
fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/ul[0]/li/a=publist_navigation_link

fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/div=CLEAR_INDEX,publist_publication_type_group

fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[0]/a=teachlist_admin_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[0]/ul[0]/li[0]/a=teachlist_admin_edit-view_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/div/div[0]/h4[0]/a=teachlist_teaching_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/div/div[1]/span[0]/a=CLEAR_INDEX,teachlist_staff_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[2]/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[2]/ul[0]/li/a=teachlist_teaching_link

fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[0]/div/div[0]/h4[0]/a=teachlist_teaching_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[0]/div/div[1]/span[0]/a=CLEAR_INDEX,teachlist_staff_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/ul[0]/li/a=teachlist_teaching_link

fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div/div/ul/li=CLEAR_INDEX

fieldset(htmlId\=node_staff_full_group_staff_awards)/div[0]/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_awards)/div[0]/div[0]/div/div[0]/ul/li=CLEAR_INDEX

fieldset(htmlId\=node_staff_full_group_staff_projects)/div[0]/div[0]=ongoing_projects
div(htmlId\=ongoing_projects)/div[0]=project_list
div(htmlId\=ongoing_projects)/div[1]=project_list
fieldset(htmlId\=node_staff_full_group_staff_projects)/div[0]/div[1]=past_projects
div(htmlId\=past_projects)/div[0]=project_list
div(htmlId\=past_projects)/div[1]=project_list

div(htmlId\=project_list)/a=project_list_admin_link
div(htmlId\=project_list)/ul[0]/li[0]/a=project_list_admin_edit-view_link
div(htmlId\=project_list)/div=CLEAR_INDEX
div(htmlId\=project_list)/div/div[0]/h4/a=project_link

#################### Research overview page
div(htmlId\=page_research_overview)/div[1]=node-research-overview
div(htmlId\=page_research_overview)/div[2]=node-research-overview

#################### Research details
div(htmlId\=page_research_project-details)/div=CLEAR_INDEX
div(htmlId\=page_research_project-details)/div/div=node-project-details
div(htmlId\=page_research_project-details)/div/ul/li=CLEAR_INDEX,project_related_publication

div(htmlId\=node-project-details)/div[0]/div/div[0]/a[0]=admin_link
div(htmlId\=node-project-details)/div[0]/div/div[0]/ul[0]/li[0]/a[0]=admin_edit-view_link

div(htmlId\=node-project-details)/div[0]/div[0]/div/div[0]/div[0]/span[0]/a=CLEAR_INDEX,research_staff_link
div(htmlId\=node-project-details)/div[0]/div/div[1]/div[0]/a=research_details_link
div(htmlId\=node-project-details)/div[0]/div[4]=node_project_details_publications
div(htmlId\=node-project-details)/div[0]/div[5]=node_project_details_publications
div(htmlId\=node-project-details)/div[0]/div[6]=node_project_details_publications
div(htmlId\=node-project-details)/div[0]/div[7]=node_project_details_publications
div(htmlId\=node_project_details_publications)/div[0]/div[0]/div[0]/div[0]/div[0]/div[0]/ul[0]/li/a=research_project_link
div(htmlId\=node_project_details_publications)/div[0]/div[0]=publist_publication_type_group
div(htmlId\=node_project_details_publications)/div[0]/div[1]=publist_publication_type_group

div(htmlId\=node_project_full_group_project_details)/div[0]/div[1]=CLEAR_INDEX
div(htmlId\=node_project_full_group_project_details)/div[3]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[4]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[5]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[6]=node_project_details_list
div(htmlId\=node_project_details_list)/div[0]=node_project_details_list2
div(htmlId\=node_project_details_list)/div[1]=node_project_details_list2
div(htmlId\=node_project_details_list2)/div[0]/ul/li=CLEAR_INDEX

#################### Teaching details
div(htmlId\=node-lectures-details)/div[0]/div[0]=lecture-details-staff
div(htmlId\=lecture-details-staff)/div=CLEAR_INDEX
div(htmlId\=lecture-details-staff)/div/div[0]/div[0]/span[0]/a=CLEAR_INDEX,teaching_staff_link

div(htmlId\=node-lectures-details)/div[0]/div[1]=lecture-details-description
div(htmlId\=node-lectures-details)/div[0]/div[2]=lecture-details-description

div(htmlId\=node-lectures-details)/div[0]/div[3]=lecture-details-files
div(htmlId\=node-lectures-details)/div[0]/div(htmlId\=node_lecture_full_group_lecture_files)=lecture-details-files
div(htmlId\=lecture-details-files)/div=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/ol[0]/li=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/ol[0]/li/span[0]/a[0]=lecture-details-file-link
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/div[0]/ol[0]/li=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/div[0]/ol[0]/li/span[0]/a[0]=lecture-details-file-link


#################### Publication details
div(htmlId\=node-publication-details)/div[0]/div[0]/div[0]/div[0]/div[0]/span[0]/a=CLEAR_INDEX,publication_author_link
div(htmlId\=node-publication-details)/div[0]/div[0]/div[1]/div[0]/div[0]/span[0]/a=CLEAR_INDEX,publication_author_link
div(htmlId\=node-publication-details)/div[0]/div[4]/div[0]/div[0]/ul[0]/li[0]/span/span[0]/a=publication_details_link
div(htmlId\=node-publication-details)/div[0]/div[6]/div[1]/div[0]/span[0]/a[0]=publication_file_link
div(htmlId\=node-publication-details)/div[0]/div[6]/h2[0]/span[0]/a[0]=publication_bibtex_link
div(htmlId\=node-publication-details)/div[0]/div[6]/div[0]/div[0]/div[0]/div[0]/pre[0]/a=publication_bibtex_details_link
div(htmlId\=node-publication-details)/div[0]/div[7]/h2[0]/span[0]/a[0]=publication_bibtex_link
div(htmlId\=node-publication-details)/div[0]/div[7]/div[0]/div[0]/div[0]/div[0]/pre[0]/a=publication_bibtex_details_link
div(htmlId\=node-publication-details)/div[2]/ul/li=CLEAR_INDEX
div(htmlId\=node-publication-details)/div[2]/ul/li/a=publication_author_link

div(htmlId\=node_publication_full_group_publication_further_infos)/div[4]/div=


#################### Common replacements
div(htmlId\=publist_publication_type_group)/ul[0]/li=CLEAR_INDEX
div(htmlId\=publist_publication_type_group)/ul[0]/li/a=publist_navigation_link
div(htmlId\=publist_publication_type_group)/ul[0]/li/span=CLEAR_INDEX
div(htmlId\=publist_publication_type_group)/ul[0]/li/span/span=CLEAR_INDEX
div(htmlId\=publist_publication_type_group)/ul[0]/li/span/span/a=publist_author_link

#################### Other pages
div(htmlId\=page_admin_nodes_recently-added)/div[1]/div=
div(htmlId\=page_staff_former_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr=CLEAR_INDEX
div(htmlId\=page_staff_former_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td=CLEAR_INDEX
```

B.2.2. Parsing Configuration for the New Website Version

```
#################### Mapping of document paths
document(path\=/awards$)=awards/overview
document(path\=/awards/index.php$)=awards/overview
document(path\=/awards/###############)=awards/details
document(path\=/certified-tester)=other/certified-tester
document(path\=/emc-academic-alliance)=other/emc-academic-alliance
document(path\=/external-academic-services/######)=staff/external-academic-services/details
document(path\=/external-academic-services-taxonomy/######)=admin/staff/external-academic-services/taxonomy
document(path\=/filter/tips)=admin/help/filter/tips
document(path\=/former-staff)=staff/former/overview
document(path\=/help/facetapi_bonus/README.txt)=admin/help/facetapi_bonus
document(path\=/how-to-find-us)=home/how-to-find-us
document(path\=/import)=admin/import
document(path\=/imprint)=other/imprint
document(path\=^/index.php$)=/
document(path\=/ingo-tributh)=CLEAR_QUERY,staff/details
document(path\=/internal-academic-services/######)=staff/internal-academic-services/details
document(path\=/jobs$)=staff/jobs/overview
document(path\=/jobs/index.php$)=staff/jobs/overview
document(path\=/jobss$)=staff/jobs/overview
document(path\=/jobss/$)=staff/jobs/overview
document(path\=/jobss/######)=staff/jobs/details
document(path\=/lectures$)=lectures/overview
document(path\=/lectures/$)=lectures/overview
document(path\=/lectures/######)=CLEAR_QUERY,lectures/details
document(path\=/news$)=CLEAR_QUERY,home/news/overview
document(path\=/news/$)=CLEAR_QUERY,home/news/overview
document(path\=/news/######)=CLEAR_QUERY,home/news/details
document(path\=/news/index.php$)=CLEAR_QUERY,home/news/overview
document(path\=/node/$)=admin/nodes/recently-added
document(path\=/node/####$)=publications/details
document(path\=/our-research$)=research/overview
document(path\=/publications$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/######)=CLEAR_QUERY,publications/details
document(path\=/publications-recent$)=CLEAR_QUERY,publications/recent/overview

document(path\=/publications/A-98-04/Report-A-98-04.ps.gz$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/BZ_DV_IS_HN_JG/main.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/BZ_HN_JG_DE_PB/trex.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/EE/Masterarbeit_SRE_Eduard_Enriquez_28072008-1.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ER_IS_JG/SAM2000RudolphEtAll.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ER_JG_PG/SDL99-Harmonization.ps.gz$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/EW_JG_ST_BZ/WSTestFramework_main.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/FBT98/FBT98.ps.gz$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/HN_BZ_JG_PB_DE/stvr_quality_assurance_for_ttcn3_test_specifications.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/HN_BZ/wrt_ecoop2007.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/HN_MB/ttcn3codesmells_testcom2007.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/IS_ZD_JG_AR/TestCom2003_UTP_Final.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/JG_ER/MSC-Survey93.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/JG/Grabowski.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/JK/bmsc_thesis_kemnade.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/MB/bisanz_mastersthesis.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/PB_ZD_JG_ME_GK_IS_PG/U2TP-CONQUEST-2004.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ST/WebServiceTestFrameworkWithTTCN-3.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/TR_AA_JM_FV/rings-gb74.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/TR_HN_JG/rings-testingGridApplication.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ZD_JG_HN_HP/jzus.pdf$)=CLEAR_QUERY,publications/all/overview
document(path\=/publications/ZD_JG_HN_HP/testcom2004_from_design_to_test_with_uml.pdf$)=CLEAR_QUERY,publications/all/overview

document(path\=/research$)=CLEAR_QUERY,research/overview
document(path\=/research/######)=research/project-details
document(path\=/search/node/)=search-results
document(path\=/semester/ws2014)=admin/taxonomy/list

document(path\=^/staff$)=CLEAR_QUERY,staff/current/overview
document(path\=^/staff/$)=CLEAR_QUERY,staff/current/overview
document(path\=^/staff/index.php$)=CLEAR_QUERY,staff/current/overview
document(path\=^/staff/#######)=CLEAR_QUERY,staff/details
document(path\=^/staff/single_staff/index.php)=CLEAR_QUERY,staff/current/overview

document(path\=/start$)=/
document(path\=/taxonomy/term/#)=admin/taxonomy/term/details
document(path\=/teaching$)=CLEAR_QUERY,teaching/overview
document(path\=/teaching-all$)=CLEAR_QUERY,teaching/overview
document(path\=/teaching/writing-and-presenting$)=teaching/writing-and-presenting
document(path\=/teaching/writing-and-presentings)=CLEAR_QUERY,teaching/overview
document(path\=/user$)=CLEAR_QUERY,user/login
document(path\=/user/login$)=CLEAR_QUERY,user/login
document(path\=/user/logout$)=CLEAR_QUERY,user/logout
document(path\=/user/password$)=CLEAR_QUERY,user/password
document(path\=/user/reset/###/########################################################)=user/password-reset
document(path\=/users/$)=admin/users/overview
document(path\=/users/admin$)=admin/users/overview
document(path\=/users/sadhatarao$)=admin/users/details
document(path\=/users/students$)=admin/users/details
document(path\=/users/vhonsel$)=admin/users/details

document(path\=/we-wish-you-all-merry-christmas-and-happy-new-year$)=/
document(path\=/we-wish-you-merry-christmas-and-happy-new-year$)=/

document(path\=/compscience/accounts/$)=CLEAR_QUERY,invalid-page
document(path\=/compscience/accounts/login/)=CLEAR_QUERY,invalid-page
document(path\=/compscience/application/)=CLEAR_QUERY,invalid-page
document(path\=/edu$)=CLEAR_QUERY,invalid-page
document(path\=/edu/$)=CLEAR_QUERY,invalid-page
document(path\=/edu/eval.php)=CLEAR_QUERY,invalid-page
document(path\=/edu/lv.php)=CLEAR_QUERY,invalid-page
document(path\=/edu/notes/$)=CLEAR_QUERY,invalid-page
document(path\=/edu/notes/index.php)=CLEAR_QUERY,invalid-page
document(path\=/favicon.ico)=CLEAR_QUERY,invalid-page
document(path\=/intranet/bibo/index.php)=CLEAR_QUERY,invalid-page
document(path\=/itis/2015/suse/accounts/login/)=CLEAR_QUERY,invalid-page
document(path\=/itis/2015/suse/application/)=CLEAR_QUERY,invalid-page
document(path\=/itis/accounts/login/)=CLEAR_QUERY,invalid-page
document(path\=/itis/accounts/password/change/)=CLEAR_QUERY,invalid-page
document(path\=/itis/accounts/password/reset/)=CLEAR_QUERY,invalid-page
document(path\=/itis/application/)=CLEAR_QUERY,invalid-page
document(path\=/itis/imprint/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/$)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/41/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/500/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/78/)=CLEAR_QUERY,invalid-page
document(path\=/itis/review/login/)=CLEAR_QUERY,invalid-page
document(path\=/itis/wizard/)=CLEAR_QUERY,invalid-page
document(path\=/lecture$)=CLEAR_QUERY,invalid-page
document(path\=/lectures/exam-ss2014)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ss2014)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ws2003)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ws2013)=CLEAR_QUERY,invalid-page
document(path\=/lectures/ws2014)=CLEAR_QUERY,invalid-page
document(path\=/ectures/software-testing-ws2014)=CLEAR_QUERY,invalid-page
document(path\=^/notes/$)=CLEAR_QUERY,invalid-page
document(path\=/notes/SS2004/Grabowski/I-Einfuehrung-6-auf-1.pdf)=CLEAR_QUERY,invalid-page
document(path\=/notes/SS2007/grabowski/0-Allgemeines-Organisatorisches-6-auf-1.pdf)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2004/$)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2004/neukirchen/$)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2004/neukirchen/case-tools_Noedler.pdf)=CLEAR_QUERY,invalid-page
document(path\=/notes/WS2008/bzeiss/51/07collectiveownership.pdf)=CLEAR_QUERY,invalid-page
document(path\=/pubs/pub_liste/pub_anzeige.php)=CLEAR_QUERY,invalid-page
document(path\=/pubs/single_pub/index.php)=CLEAR_QUERY,invalid-page
document(path\=/research/main_projects/projects/single_projects/index.php)=CLEAR_QUERY,invalid-page
document(path\=/se$)=CLEAR_QUERY,invalid-page
document(path\=/sites/all/themes/samara/styles/custom.css)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/pages/)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/DissertationGrabowski.pdf)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/main1.pdf)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/publications/PhD_wafi.pdf)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/pictures/$)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/pictures/IMG_0466.JPG%3Fitok%3DJSPAGeSr)=CLEAR_QUERY,invalid-page
document(path\=/sites/default/files/styles/thumbnail/public/staff/pictures/verena10_0.JPG%3Fitok%3Dgm4kHYS0)=CLEAR_QUERY,invalid-page
document(path\=/theses/criteria.php)=CLEAR_QUERY,invalid-page
document(path\=/theses/index.php)=CLEAR_QUERY,invalid-page
document(path\=/trac$)=CLEAR_QUERY,invalid-page
document(path\=/trac/wiki/TracIni)=CLEAR_QUERY,invalid-page
document(path\=/ina-schieferdecker)=CLEAR_QUERY,invalid-page
document(path\=/ingo-tributh)=CLEAR_QUERY,invalid-page
document(path\=^/node/)=CLEAR_QUERY,invalid-page
document(path\=/publica%20tions/)=CLEAR_QUERY,invalid-page
document(path\=/semester/ws2014)=CLEAR_QUERY,invalid-page
document(path\=/svn/phd/fglaser)=CLEAR_QUERY,invalid-page
document(path\=/swetrac)=CLEAR_QUERY,invalid-page
document(path\=/swz.png)=CLEAR_QUERY,invalid-page
document(path\=^/system/files/)=CLEAR_QUERY,invalid-page
document(path\=/theses/criteria.php)=CLEAR_QUERY,invalid-page


#################### Mapping major page contents
document(path\=/$)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=main)/div(htm\
    lId\=page)=page_root
document(path\=admin/help/facetapi_bonus)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/di\
    v(htmlId\=main)/div(htmlId\=page)=page_admin_help_facetapi_bonus
document(path\=admin/help/filter/tips)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(h\
    tmlId\=main)/div(htmlId\=page)=page_admin_help_filter_tips
document(path\=admin/import)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=mai\
    n)/div(htmlId\=page)=page_admin_import
document(path\=admin/nodes/recently-added)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/d\
    iv(htmlId\=main)/div(htmlId\=page)=page_admin_nodes_recently-added
document(path\=admin/staff/external-academic-services/taxonomy)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(ht\
    mlId\=main-wrapper)/div(htmlId\=main)/div(htmlId\=page)=page_admin_staff_external-academic-services_taxonomy
document(path\=admin/taxonomy/list)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_admin_taxonomy_list
document(path\=admin/taxonomy/term/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/\
    div(htmlId\=main)/div(htmlId\=page)=page_admin_taxonomy_term_details
document(path\=admin/users/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_admin_users_details
document(path\=admin/users/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htm\
    lId\=main)/div(htmlId\=page)=page_admin_users_overview
document(path\=awards/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=m\
    ain)/div(htmlId\=page)=page_awards_details
document(path\=awards/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=\
    main)/div(htmlId\=page)=page_awards_overview
document(path\=home/how-to-find-us)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_home_how-to-find-us
document(path\=home/news/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_home_news_details
document(path\=home/news/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlI\
    d\=main)/div(htmlId\=page)=page_home_news_overview
document(path\=lectures/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\\
    =main)/div(htmlId\=page)=page_lectures_details
document(path\=lectures/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_lectures_overview
document(path\=other/certified-tester)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(h\
    tmlId\=main)/div(htmlId\=page)=page_other_certified-tester
document(path\=other/emc-academic-alliance)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/\
    div(htmlId\=main)/div(htmlId\=page)=page_other_emc-academic-alliance
document(path\=other/imprint)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=ma\
    in)/div(htmlId\=page)=page_other_imprint
document(path\=publications/all/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/di\
    v(htmlId\=main)/div(htmlId\=page)=page_publications_all_overview
document(path\=publications/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htm\
    lId\=main)/div(htmlId\=page)=page_publications_details
document(path\=publications/recent/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)\
    /div(htmlId\=main)/div(htmlId\=page)=page_publications_recent_overview
document(path\=research/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_research_overview
document(path\=research/project-details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div\
    (htmlId\=main)/div(htmlId\=page)=page_research_project-details
document(path\=search-results)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=m\
    ain)/div(htmlId\=page)=page_search-results
document(path\=staff/current/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(h\
    tmlId\=main)/div(htmlId\=page)=page_staff_current_overview
document(path\=staff/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=ma\
    in)/div(htmlId\=page)=page_staff_details
document(path\=staff/external-academic-services/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=m\
    ain-wrapper)/div(htmlId\=main)/div(htmlId\=page)=page_staff_external-academic-services_details
document(path\=staff/former/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(ht\
    mlId\=main)/div(htmlId\=page)=page_staff_former_overview
document(path\=staff/internal-academic-services/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=m\
    ain-wrapper)/div(htmlId\=main)/div(htmlId\=page)=page_staff_internal-academic-services_details
document(path\=staff/jobs/details)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlI\
    d\=main)/div(htmlId\=page)=page_staff_jobs_details
document(path\=staff/jobs/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_staff_jobs_overview
document(path\=teaching/overview)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\
    \=main)/div(htmlId\=page)=page_teaching_overview
document(path\=teaching/writing-and-presenting)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapp\
    er)/div(htmlId\=main)/div(htmlId\=page)=page_teaching_writing-and-presenting
document(path\=user/login)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=main)\
    /div(htmlId\=page)=page_user_login
document(path\=user/logout)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=main\
    )/div(htmlId\=page)=page_user_logout
document(path\=user/password$)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(htmlId\=m\
    ain)/div(htmlId\=page)=page_user_password
document(path\=user/password-reset)/html/body/div(htmlId\=main-columns-wrapper)/div(htmlId\=main-columns)/div(htmlId\=main-wrapper)/div(html\
    Id\=main)/div(htmlId\=page)=page_user_password-reset

#################### Overall
body
div(htmlId\=logo)/a(htmlId\=logo_home)/img=logo_img
body[0]/div(htmlId\=skip-link)/a=skip_link
```

```
#################### Start page
div(htmlId\=page)/div/ul[0]/li[0]/a[0]=admin_view
div(htmlId\=page)/div/ul[0]/li[1]/a[0]=admin_edit
div(htmlId\=page)/div/ul[0]/li[2]/a[0]=admin_log

div(htmlId\=node-startpage)/div[0]/div[0]/div[0]/div[0]/div[0]/a[0]=start_staff_link
div(htmlId\=node-startpage)/div[0]/div[0]/div[0]/div[0]/div[1]/a[0]=start_mail

div(htmlId\=node-startpage)/div[0]/div[1]/div[0]/div=CLEAR_INDEX
div(htmlId\=node-startpage)/div[0]/div[1]/div[0]/div/div[0]/span[0]/a[0]=start_newslink

#################### Menu
div(htmlId\=navigation)/ul[0]/li[0]/a=menu_home
div(htmlId\=navigation)/ul[0]/li[0]/ul[0]/li[0]/a=menu_how-to-find-us

div(htmlId\=navigation)/ul[0]/li[1]/a=menu_staff
div(htmlId\=navigation)/ul[0]/li[1]/ul[0]/li[0]/a=menu_staff_current
div(htmlId\=navigation)/ul[0]/li[1]/ul[0]/li[1]/a=menu_staff_jobs
div(htmlId\=navigation)/ul[0]/li[1]/ul[0]/li[2]/a=menu_staff_former

div(htmlId\=navigation)/ul[0]/li[2]/a=menu_research

div(htmlId\=navigation)/ul[0]/li[3]/a=menu_publications
div(htmlId\=navigation)/ul[0]/li[3]/ul[0]/li[0]/a=menu_publications_all
div(htmlId\=navigation)/ul[0]/li[3]/ul[0]/li[1]/a=menu_publications_recent

div(htmlId\=navigation)/ul[0]/li[4]/a=menu_awards

div(htmlId\=navigation)/ul[0]/li[5]/a=menu_teaching
div(htmlId\=navigation)/ul[0]/li[5]/ul[0]/li[0]/a=menu_teaching_lectures
div(htmlId\=navigation)/ul[0]/li[5]/ul[0]/li[1]/a=menu_teaching_writing-and-presenting

#################### Toolbar
ul(htmlId\=toolbar-home)/li[0]/a[0]=toolbar_home
ul(htmlId\=toolbar-user)/li[0]/a[0]=toolbar_userlink
ul(htmlId\=toolbar-user)/li[1]/a[0]=toolbar_logout
div(htmlId\=toolbar)/div[0]/a[0]=hidden_link
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[0]/a[0]=toolbar_add-content
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[1]/a[0]=toolbar_find-content
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[2]/a[0]=toolbar_install-module
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[3]/a[0]=toolbar_taxonomy
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[4]/a[0]=toolbar_feeds
div(htmlId\=toolbar)/div[1]/div[0]/ul[0]/li[5]/a[0]=toolbar_content-types


#################### Other common stuff
div(htmlId\=closure)/div(htmlId\=info)/ul(htmlId\=navlist2)/li[0]/a[0]=closure_imprint
div(htmlId\=closure)/div(htmlId\=info)/ul(htmlId\=navlist2)/li[1]/a[0]=closure_login

div(htmlId\=branding)/div(htmlId\=logo)/a=logo_home
div(htmlId\=branding)/div(htmlId\=name-and-slogan)/div(htmlId\=site-name)/a=slogan_home
div(htmlId\=branding)/div(htmlId\=name-and-slogan)/div(htmlId\=site-slogan)/a[0]=link_ifi
div(htmlId\=branding)/div(htmlId\=name-and-slogan)/div(htmlId\=site-slogan)/a[1]=link_unigoe
div(htmlId\=logo-title)/a=logo_home
div(htmlId\=name-and-slogan)/h1(htmlId\=site-name)/a=slogan_home

#################### Sidebars
div(htmlId\=block-facetapi-dii7dzt2w202fmxdxuz3k6zkhahv17pd)=block-facetapi
div(htmlId\=block-facetapi-ba870gqlmgxg3kakttxrq2y1paet88yj)=block-facetapi
div(htmlId\=block-facetapi-8fztyvn1a9ayhbbsvwslsqun2oz84k0z)=block-facetapi
div(htmlId\=block-facetapi-jsbyxjgad7ybezm3vzrmtaitmhzm21ru)=block-facetapi
div(htmlId\=block-facetapi-08o0hn2r9xl8lxj01sb93yxggnz6cv5i)=block-facetapi
div(htmlId\=block-facetapi-0h5tt9na1lb6gstmitjntvydp9xjr481)=block-facetapi
div(htmlId\=block-facetapi)/div=CLEAR_INDEX
div(htmlId\=block-facetapi)/div/div[0]/a[0]=filter_showmore_link
div(htmlId\=block-facetapi)/a=block-facetapi_configure_link
div(htmlId\=block-facetapi)/div/a=block-facetapi_configure_link
div(htmlId\=block-facetapi)/div/ul/li[0]/a=block-facetapi_configure_display
div(htmlId\=block-facetapi)/div/ul/li[1]/a=block-facetapi_configure_dependencies
div(htmlId\=block-facetapi)/div/ul/li[2]/a=block-facetapi_configure_filters
div(htmlId\=block-facetapi)/div/ul/li[3]/a=block-facetapi_configure_block

div(htmlId\=block-current-search-standard)/div=CLEAR_INDEX
div(htmlId\=block-current-search-standard)/div/a=block-current-search_configure_link
div(htmlId\=block-current-search-standard)/div/ul/li[0]/a=block-current-search_configure_items
div(htmlId\=block-current-search-standard)/div/ul/li[1]/a=block-current-search_configure_block
div(htmlId\=block-views-projects-block-2)=block_ongoing_projects
div(htmlId\=block_ongoing_projects)/h2/a=block_project_header
div(htmlId\=block_ongoing_projects)/div[0]=block_project_list
div(htmlId\=block_ongoing_projects)/div[1]=block_project_list
div(htmlId\=block-views-projects-block-1)=block_past_projects
div(htmlId\=block_past_projects)/h2/a=block_project_header
div(htmlId\=block_past_projects)/div[0]=block_project_list
div(htmlId\=block_past_projects)/div[1]=block_project_list

div(htmlId\=block_project_list)/a=block_admin_link
div(htmlId\=block_project_list)/ul[0]/li[0]/a=block_admin_edit-view_link
div(htmlId\=block_project_list)/ul[0]/li[1]/a=block_admin_configure-view_link
div(htmlId\=block_project_list)/div[0]/div[0]/ul[0]/li=CLEAR_INDEX
div(htmlId\=block_project_list)/div[0]/div[0]/ul[0]/li/div[0]/span[0]/a[0]=project_link
div(htmlId\=block_project_list)/div[0]/div[0]/div[0]/ul[0]/li=CLEAR_INDEX
div(htmlId\=block_project_list)/div[0]/div[0]/div[0]/ul[0]/li/div[0]/span[0]/a[0]=project_link

div(htmlId\=block-views-news-block-1)/div=CLEAR_INDEX
div(htmlId\=block-views-news-block-1)/div/a=block_admin_link
div(htmlId\=block-views-news-block-1)/div/ul[0]/li[0]/a=block_admin_edit-view_link
div(htmlId\=block-views-news-block-1)/div/ul[0]/li[1]/a=block_admin_configure-view_link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/ul/li=CLEAR_INDEX,news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/div=CLEAR_INDEX
div(htmlId\=block-views-news-block-1)/div/div[0]/div/div/ul/li=CLEAR_INDEX,news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/div/div[0]/h3/a=news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/div/div[0]/h4/a=news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/div/div[0]/p/a=news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/div/p/a=news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div=CLEAR_INDEX
div(htmlId\=block-views-news-block-1)/div/div[0]/div/a=news-link
div(htmlId\=block-views-news-block-1)/div/div[0]/div/p/a=news-link

div(htmlId\=block-views-teaching-block-1)/div=CLEAR_INDEX
div(htmlId\=block-views-teaching-block-1)/div/a=block_admin_link
div(htmlId\=block-views-teaching-block-1)/div/ul[0]/li[0]/a=block_admin_edit-view_link
div(htmlId\=block-views-teaching-block-1)/div/ul[0]/li[1]/a=block_admin_configure-view_link
div(htmlId\=block-views-teaching-block-1)/div/div[0]/div/ul=CLEAR_INDEX
div(htmlId\=block-views-teaching-block-1)/div/div[0]/div/ul/li=CLEAR_INDEX
div(htmlId\=block-views-teaching-block-1)/div/div[0]/div/ul/li/a=teaching-link
div(htmlId\=block-views-teaching-block-1)/div/div[0]/div=CLEAR_INDEX
div(htmlId\=block-views-teaching-block-1)/div/div[0]/div/a=teaching-link
div(htmlId\=block-views-teaching-block-1)/div/div[0]/div/p/a=teaching-link

##################### Login page
div(htmlId\=page_user_login)/div=CLEAR_INDEX
div(htmlId\=page_user_login)/div/ul/li[0]/a=login_edit-submit_or_view-link
div(htmlId\=page_user_login)/div/ul/li[1]/a=login_request-new-password_or_edit
div(htmlId\=page_user_login)/div/ul/li[2]/a=login_shortcuts

##################### Root page
div(htmlId\=page_root)/div=CLEAR_INDEX

##################### News details page
div(htmlId\=page_home_news_details)/div=CLEAR_INDEX
div(htmlId\=page_home_news_details)/div/div=node-news-details

##################### News overview page
div(htmlId\=page_home_news_overview)/div[1]/div[0]/div/div=node-news-list-entry
div(htmlId\=page_home_news_overview)/div[1]/div[0]/div[1]/ul/li=CLEAR_INDEX
div(htmlId\=page_home_news_overview)/div[1]/div[0]/div[1]/ul/li/a=pager-link

##################### Lectures details page
div(htmlId\=page_lectures_details)/div=CLEAR_INDEX
div(htmlId\=page_lectures_details)/div/div=node-lectures-details

##################### External Academic Service details page
div(htmlId\=page_staff_external-academic-services_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_external-academic-services_details)/div/div=node-external-academic-services-details
div(htmlId\=page_staff_external-academic-services_details)/div/ul/li=CLEAR_INDEX

##################### External Academic Service details page
div(htmlId\=page_admin_staff_external-academic-services_taxonomy)/div[1]/div=node-external-academic-services-details

##################### Internal Academic Service details page
div(htmlId\=page_staff_internal-academic-services_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_internal-academic-services_details)/div/div=node-internal-academic-services-details
div(htmlId\=page_staff_internal-academic-services_details)/div/ul/li=CLEAR_INDEX
```

```
##################### Internal Academic Service details page
div(htmlId\=page_admin_staff_internal-academic-services_taxonomy)/div[1]/div=node-internal-academic-services-details

##################### Staff details page
div(htmlId\=page_staff_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_details)/div/div=node-staff-details

##################### Publications details page
div(htmlId\=page_publications_details)/div=CLEAR_INDEX
div(htmlId\=page_publications_details)/div/div=node-publication-details

##################### Awards page
div(htmlId\=page_awards_overview)/div[1]/div[0]/div[0]/div[0]/ul[0]/li=award-description

##################### Search results page
div(htmlId\=page_search-results)/div[1]/ol/li=CLEAR_INDEX
div(htmlId\=page_search-results)/div[1]/ol/li/h3/a=search_result_link

##################### Staff list
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr=CLEAR_INDEX
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td=CLEAR_INDEX
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[0]/div[0]/a[0]=stafflist_imglink
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[0]/div[0]/a[0]/img=stafflist_img
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[1]/span[0]/a[0]=stafflist_namelink
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[3]/div[0]/span=stafflist_telno
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[3]/div[0]/span/span=
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[3]/div[0]/span/span/span=
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[4]/div[0]/a[0]=stafflist_email
div(htmlId\=page_staff_current_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td/div[4]/div[0]/p[0]/a[0]=stafflist_email

##################### Jobs details page
div(htmlId\=page_staff_jobs_details)/div=CLEAR_INDEX
div(htmlId\=page_staff_jobs_details)/div/div=node-jobs-details

##################### Jobs overview page
div(htmlId\=page_staff_jobs_overview)/div[1]/div[0]/div[0]/div=CLEAR_INDEX

##################### Teaching list
div(htmlId\=page_teaching_overview)/div[1]=node-teaching-overview
div(htmlId\=page_teaching_overview)/div[2]=node-teaching-overview
div(htmlId\=page_teaching_overview)/div[3]=node-teaching-overview
div(htmlId\=node-teaching-overview)/div[0]/div[0]/div=course-entry
div(htmlId\=course-entry)/div[0]/h4/a=course-link
div(htmlId\=course-entry)/div[1]/span=CLEAR_INDEX
div(htmlId\=course-entry)/div[1]/span/a=course-staff-link
div(htmlId\=node-teaching-overview)/div[0]/div[1]/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-teaching-overview)/div[0]/div[1]/ul[0]/li/a=pager-link

ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-semester)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-semester)/li/a=teachlist_term_filter
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-staff)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-staff)/li/a=teachlist_staff_filter
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-type)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apilecture-index-block-field-lecture-type)/li/a=teachlist_type_filter

##################### Publication list
div(htmlId\=page_publications_all_overview)/div=CLEAR_INDEX
div(htmlId\=page_publications_all_overview)/div/div=node-publication-overview
div(htmlId\=page_publications_all_overview)/div/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-publication-overview)/div[1]/div=publication-type-group
div(htmlId\=node-publication-overview)/div[2]/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-publication-overview)/div[2]/ul[0]/li/a=pager-link

div(htmlId\=page_publications_recent_overview)/div[1]=publication-list
div(htmlId\=page_publications_recent_overview)/div[3]=publication-list
div(htmlId\=publication-list)/div[0]/div[0]/div=publication-type-group

ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-year)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-year)/li/a=publist_year_filter
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-author)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-author)/li/a=publist_author_filter
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-doc-type)/li=CLEAR_INDEX
ul(htmlId\=facetapi-facet-search-apipublication-index-block-field-pub-doc-type)/li/a=publist_doc-type_filter

##################### Staff details
div(htmlId\=node-staff-details)/div[1]/div[0]/div[0]/div[0]/img=staff_image

div(htmlId\=node-staff-details)/div[2]/div=CLEAR_INDEX
div(htmlId\=node-staff-details)/div[2]/div/div=CLEAR_INDEX
div(htmlId\=node-staff-details)/div[2]/div/div/div=CLEAR_INDEX
div(htmlId\=node-staff-details)/div[2]/div/div/div/div[0]=staff_academic_degree
div(htmlId\=node-staff-details)/div[2]/div/div/div/div[1]=staff_position
div(htmlId\=node-staff-details)/div[2]/div/div/div/span[0]=staff_email
span(htmlId\=staff_email)/a[0]=staff_email_link
div(htmlId\=node-staff-details)/div[2]/div/div/div/div[2]=staff_phone
div(htmlId\=node-staff-details)/div[2]/div/div/div/div[3]=staff_room
div(htmlId\=node-staff-details)/div[2]/div/div/div/div[4]=staff_officehours

div(htmlId\=node-staff-details)/div[3]/div[0]/div[0]/ul[0]/li=CLEAR_INDEX
div(htmlId\=node-staff-details)/div[3]/div[0]/div[0]/ul[0]/li/a=staff_verticaltab_selection

fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/a=publist_admin_link
fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/ul[0]/li/a=publist_navigation_link

fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_publications)/div[0]/div[0]/div/div=CLEAR_INDEX,publist_publication_type_group

fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/a=teachlist_admin_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/ul[0]/li/a=pager_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/h3=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/div/div[0]/h4[0]/a=teachlist_teaching_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div/div/div[1]/span[0]/a=CLEAR_INDEX,teachlist_staff_link
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_teaching)/div[0]/div[0]/div[1]/ul[0]/li/a=teachlist_teaching_link

fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div/ul[0]/li=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_as)/div[0]/div/div/div/ul/li=CLEAR_INDEX

fieldset(htmlId\=node_staff_full_group_staff_awards)/div[0]/div[0]/div=CLEAR_INDEX
fieldset(htmlId\=node_staff_full_group_staff_awards)/div[0]/div[0]/div/div[0]/ul/li=CLEAR_INDEX

fieldset(htmlId\=node_staff_full_group_staff_projects)/div[0]/div[0]=ongoing_projects
div(htmlId\=ongoing_projects)/div[0]=project_list
div(htmlId\=ongoing_projects)/div[1]=project_list
div(htmlId\=ongoing_projects)/div[2]/ul/li=CLEAR_INDEX
div(htmlId\=ongoing_projects)/div[2]/ul/li/a=pager_link
fieldset(htmlId\=node_staff_full_group_staff_projects)/div[0]/div[1]=past_projects
div(htmlId\=past_projects)/div[0]=project_list
div(htmlId\=past_projects)/div[1]=project_list
div(htmlId\=past_projects)/div[2]/ul/li=CLEAR_INDEX
div(htmlId\=past_projects)/div[2]/ul/li/a=pager_link

div(htmlId\=project_list)/a=project_list_admin_link
div(htmlId\=project_list)/ul[0]/li[0]/a=project_list_admin_edit-view_link
div(htmlId\=project_list)/div=CLEAR_INDEX
div(htmlId\=project_list)/div/div[0]/h4/a=project_link

#################### Research overview page
div(htmlId\=page_research_overview)/div[1]=node-research-overview
div(htmlId\=page_research_overview)/div[2]=node-research-overview
div(htmlId\=page_research_overview)/div[3]=node-research-overview
div(htmlId\=page_research_overview)/div[4]=node-research-overview

#################### Research details
div(htmlId\=page_research_project-details)/div=CLEAR_INDEX
div(htmlId\=page_research_project-details)/div/div=node-project-details
div(htmlId\=page_research_project-details)/div/ul/li=CLEAR_INDEX,project_related_publication

div(htmlId\=node-project-details)/div[0]/div/div[0]/a[0]=admin_link
div(htmlId\=node-project-details)/div[0]/div/div[0]/ul[0]/li[0]/a[0]=admin_edit-view_link

div(htmlId\=node-project-details)/div[0]/div[0]/div/div[0]/div[0]/span[0]/a=CLEAR_INDEX,research_staff_link
div(htmlId\=node-project-details)/div[0]/div/div[1]/div[0]/a=research_details_link
div(htmlId\=node-project-details)/div[0]/div[4]=node_project_details_publications
div(htmlId\=node-project-details)/div[0]/div[5]=node_project_details_publications
div(htmlId\=node-project-details)/div[0]/div[6]=node_project_details_publications
div(htmlId\=node-project-details)/div[0]/div[7]=node_project_details_publications
div(htmlId\=node_project_details_publications)/div[0]/div[0]/div[0]/div[0]/div[0]/ul[0]/li/a=research_project_link
div(htmlId\=node_project_details_publications)/div[0]/div[0]=publist_publication_type_group
div(htmlId\=node_project_details_publications)/div[0]/div[1]=publist_publication_type_group

div(htmlId\=node_project_full_group_project_details)/div[0]=project_staff
div(htmlId\=node_project_full_group_project_details)/div[1]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[2]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[3]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[4]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[5]=node_project_details_list
div(htmlId\=node_project_full_group_project_details)/div[6]=node_project_details_list
div(htmlId\=project_staff)/div[0]/div[0]/div[0]/span/a=project_staff_link
div(htmlId\=node_project_details_list)/div[0]=node_project_details_list2
div(htmlId\=node_project_details_list)/div[1]=node_project_details_list2
div(htmlId\=node_project_details_list2)/div[0]/ul/li=CLEAR_INDEX,publication-entry
div(htmlId\=node_project_details_list2)/div[0]/div[0]/div[0]/div[0]/ul/li=CLEAR_INDEX
div(htmlId\=node_project_details_list2)/div[0]/div[0]/div[0]/div[0]/ul/li/a=related_project_link

#################### Teaching details
div(htmlId\=node-lectures-details)/div[0]/div[0]=lecture-details-staff
div(htmlId\=lecture-details-staff)/div=CLEAR_INDEX
div(htmlId\=lecture-details-staff)/div/div=CLEAR_INDEX
div(htmlId\=lecture-details-staff)/div/div/div[0]/span[0]/a=CLEAR_INDEX,teaching_staff_link
div(htmlId\=lecture-details-staff)/div/div/a=CLEAR_INDEX,teaching_staff_link
div(htmlId\=lecture-details-staff)/div/div/p/a=CLEAR_INDEX,teaching_staff_link
div(htmlId\=lecture-details-staff)/div/div/ul/li=CLEAR_INDEX
div(htmlId\=lecture-details-staff)/div/div/ul/li/a=teaching_staff_link

div(htmlId\=node-lectures-details)/div[0]/div[1]=lecture-details-description
div(htmlId\=node-lectures-details)/div[0]/div[2]=lecture-details-description

div(htmlId\=node-lectures-details)/div[0]/div[3]=lecture-details-files
div(htmlId\=node-lectures-details)/div[0]/div(htmlId\=node_lecture_full_group_lecture_files)=lecture-details-files
div(htmlId\=lecture-details-files)/div=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/ol[0]/li=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/ol[0]/li/span[0]/a[0]=lecture-details-file-link
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/div[0]/ol[0]/li=CLEAR_INDEX
div(htmlId\=lecture-details-files)/div/div/div/div[0]/div[0]/div[0]/ol[0]/li/span[0]/a[0]=lecture-details-file-link


#################### Publication details
div(htmlId\=node-publication-details)/div[0]/div[0]/div=CLEAR_INDEX
div(htmlId\=node-publication-details)/div[0]/div[0]/div/div[0]/div[0]/span[0]/a=CLEAR_INDEX,publication_author_link
div(htmlId\=node-publication-details)/div[0]/div[4]/div[0]/div[0]/ul[0]/li[0]/span/span[0]/a=publication_details_link
div(htmlId\=node-publication-details)/div[0]/div[6]/div[1]/div[0]/span[0]/a[0]=publication_file_link
div(htmlId\=node-publication-details)/div[0]/div[6]/h2[0]/span[0]/a[0]=publication_bibtex_link
div(htmlId\=node-publication-details)/div[0]/div[6]/div[0]/div[0]/div[0]/div[0]/pre[0]/a=publication_bibtex_details_link
div(htmlId\=node-publication-details)/div[0]/div[7]/h2[0]/span[0]/a[0]=publication_bibtex_link
div(htmlId\=node-publication-details)/div[0]/div[7]/div[0]/div[0]/div[0]/div[0]/pre[0]/a=publication_bibtex_details_link
div(htmlId\=node-publication-details)/div[1]=publication_authors
div(htmlId\=node-publication-details)/div[2]=publication_authors
div(htmlId\=publication_authors)/ul/li=CLEAR_INDEX
div(htmlId\=publication_authors)/ul/li/a=publication_author_link
div(htmlId\=publication_authors)/div=CLEAR_INDEX
div(htmlId\=publication_authors)/div/ul/li=CLEAR_INDEX
div(htmlId\=publication_authors)/div/ul/li/a=publication_author_link
div(htmlId\=publication_authors)/div/ul/li/div/span/a=publication_author_link
div(htmlId\=publication_authors)/div/ul/li/span/em/a=publication_author_link
div(htmlId\=publication_authors)/div/ul/li/span/span/a=publication_author_link

div(htmlId\=node_publication_full_group_publication_further_infos)/div[0]=publication_document_type
div(htmlId\=node_publication_full_group_publication_further_infos)/div[1]=publication_other_info
div(htmlId\=node_publication_full_group_publication_further_infos)/div[2]=publication_other_info
div(htmlId\=node_publication_full_group_publication_further_infos)/div[3]=publication_other_info
div(htmlId\=node_publication_full_group_publication_further_infos)/div[4]=publication_other_info
div(htmlId\=node_publication_full_group_publication_further_infos)/div[5]=publication_other_info

div(htmlId\=publication_other_info)/div=CLEAR_INDEX
div(htmlId\=publication_other_info)/div/div[0]/span[0]/span[0]/div[0]=publication_related_projects
div(htmlId\=publication_other_info)/div/div[0]/div[0]/div[0]/div[0]=publication_related_projects
div(htmlId\=publication_related_projects)/ul/li=CLEAR_INDEX
div(htmlId\=publication_related_projects)/ul/li/a=publication_related_project_link


#################### Common replacements
div(htmlId\=publist_publication_type_group)/ul[0]/li=CLEAR_INDEX,publication-entry
div(htmlId\=publication-type-group)/ul/li=CLEAR_INDEX,publication-entry
```

```
li(htmlId\=publication-entry)/div[0]/span/a=author-link
li(htmlId\=publication-entry)/span[0]/em/a=publication-link
li(htmlId\=publication-entry)/span[1]/span[0]/a=publication-medium-link

div(htmlId\=sidebar-first)=sidebar
div(htmlId\=sidebar-second)=sidebar

#################### Other pages
div(htmlId\=page_admin_nodes_recently-added)/div[1]/div=
div(htmlId\=page_staff_former_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr=CLEAR_INDEX
div(htmlId\=page_staff_former_overview)/div[1]/div[0]/div[0]/table[0]/tbody[0]/tr/td=CLEAR_INDEX


div(htmlId\=page_user_password)/div[1]=user_password_form
div(htmlId\=page_user_password)/div[2]=user_password_form

div(htmlId\=page_other_imprint)/div[1]=CLEAR_INDEX
div(htmlId\=page_other_imprint)/div[2]=CLEAR_INDEX

div(htmlId\=page_awards_details)/div[1]=CLEAR_INDEX
div(htmlId\=page_awards_details)/div[2]=CLEAR_INDEX

div(htmlId\=page_staff_current_overview)/div[1]=current_staff_table
div(htmlId\=page_staff_current_overview)/div[2]=current_staff_table
div(htmlId\=page_staff_current_overview)/div[3]=current_staff_table
```

C. Optimization for the Generation of Task Trees

A challenge for the implementation of the sequence detection described in Section 4.4.2 was the determination of n-grams in task instance lists that occur multiple times. The reason for this is, that for large amounts of input data, many different n-grams exist. Moreover, these n-grams may occur at many diverse positions in the task instance lists. To handle this, we utilized a data structure provided by AutoQUEST, which is called a *trie* [108]. A trie is a tree structure from natural language processing, which usually represents n-grams of character combinations and the number of their occurrences in texts. In our work, we use it to represent n-grams of task instances. A trie has a root node, which represents the empty n-gram, i.e., an n-gram with $n = 0$. The children of the root node are n-grams of length one, the grandchildren n-grams of length two, the great-grandchildren n-grams of length three, and so on. The n-gram represented by a node is determined by the path of the node in the trie starting from the root node. An example of a trie is shown in Figure C.1b. It was generated for the task instance list in Figure C.1a. The nodes of the trie contain a reference

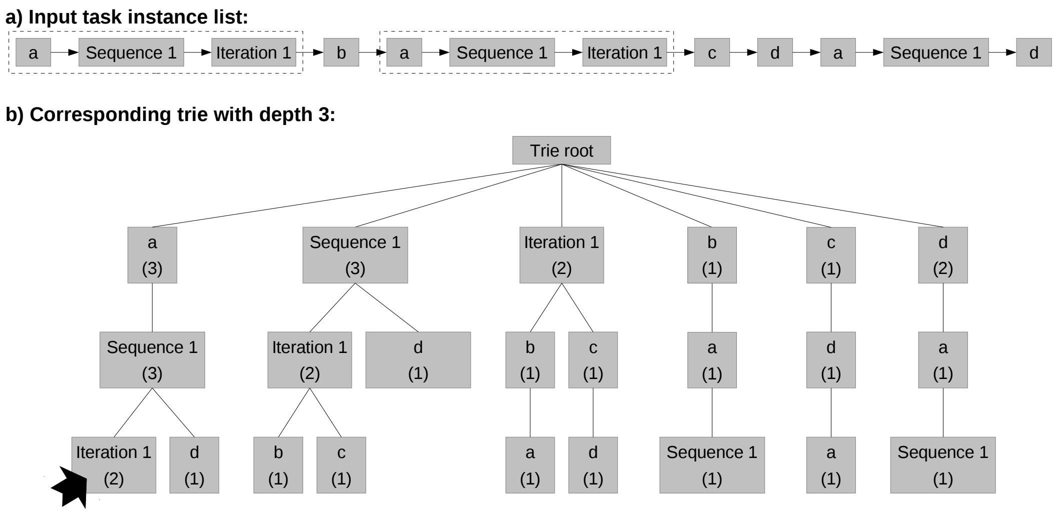


Figure C.1.: Example of a trie (b) generated based on a task instance list (a).

to the action or task that is the last element of the represented n-gram. Moreover, the nodes contain the number of occurrences of the depicted n-gram. For example, the node in the trie marked with the black arrow (Figure C.1b) represents the n-gram surrounded with the dotted boxes (Figure C.1a). This n-gram occurs twice.

A trie has a certain depth. The depth defines the maximum length of the n-grams represented by the trie. The depth of the trie in Figure C.1b is three, as the longest representable n-gram has a length of three.

For the sequence detection, we directly determine from the try the n-gram of the task instance list, that has a minimum length of two and occurs most often. For example, the trie in Figure C.1b shows, that the n-gram *{a, Sequence 1}* occurs three times and is, therefore, the single n-gram to be considered for the sequence detection.

When calculating a trie for a task instance list, we only calculate it with a certain depth. If the longest n-gram that occurs most often is as long as the depth of the trie, then there may be an even longer n-gram that occurs the same amount of times. In this case, we drop the trie and create a new one with a larger depth. We repeat the trie creation with larger depths until the trie is able to identify the longest n-gram that occurs most often. As there may be several n-grams matching this criterion, the subsequent choosing process will determine, which of these n-grams need to be handled in the sequence detection.

D. Transformation of Task Models to Other Standards

As mentioned in Section 3.4, the task trees generated by our approach are one of many existing variants of task modeling. There are other techniques utilizing tree structures, e.g., ConcurTaskTrees [53, 25], that could also be used as format for task trees in our approach. ConcurTaskTrees are the basis for several usability evaluation methods. For ConcurTask-Trees, there is a tool called ConcurTaskTrees Environment, which allows for modeling of task trees as well as their visualization [109]. In addition, the execution of task trees can be simulated. To support the utilization of our task trees in other usability evaluation approaches, as well as to be able to utilizes the visualization and simulation capabilities of the ConcurTaskTrees Environment, we implemented a transformation of our task tree structures to ConcurTaskTrees. This transformation is described in this subsection. We start the description by introducing the required concepts of ConcurTaskTrees. Afterwards, we show how we mapped our task trees to these concepts. Finally, we show an example of a transformed task tree visualized with the ConcurTaskTrees Environment.

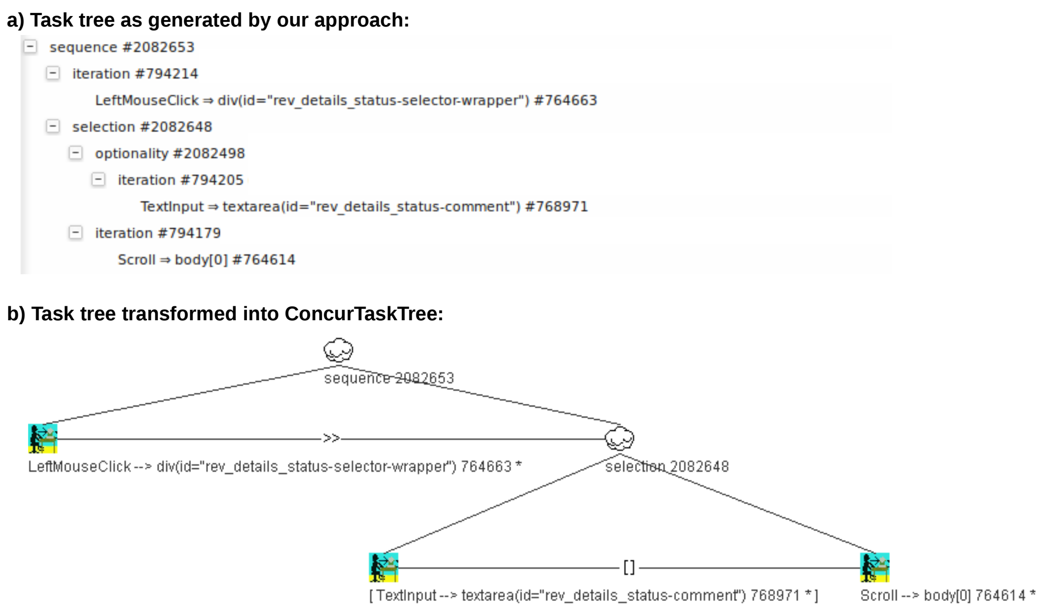
ConcurTaskTrees are task trees with a structure similar to the task trees generated by our approach. In ConcurTaskTrees, any tree node represents a task. A task can have one of four different types depending on its semantics. These four types are *abstraction task*, *interaction task*, *application task*, and *user task*. Abstraction tasks help to structure a task tree into logical subparts, which is similar to parent nodes in our approach. Interaction tasks are actions a user performs on the system, which corresponds to the leaf nodes in our task trees. Application tasks describe tasks fully executed by the system, and user tasks are fully executed by the user without interacting with the system. The latter two concepts are not considered by our task trees.

To define temporal relationships, ConcurTaskTrees utilize operators between the children of a task. For example, there is an operator called *enabling*, which defines for two children c_1 and c_2, that c_2 is enabled by c_1. This means that c_2 is executed after c_1. Furthermore, there is an operator called *choice* defining, that only one of two children connected with the operator can be performed. If a task has more than two children, different operators between the different children pairs can be used. In addition, ConcurTaskTrees allow to define for a child if it is repeated multiple times and if it is optional. To map our task tree structures to ConcurTaskTrees, we applied the following rules:

- Sequences in our task trees become abstract tasks in ConcurTaskTrees, whose children are connected using only the enabling operator.

- Selections in our task trees become abstract tasks in ConcurTaskTrees, whose children are connected using only the choice operator.
- Iterations or optionals in our task trees do not become nodes in ConcurTaskTrees. Instead their single child is added at the position of the iteration or optional in the parent task and marked as repeating or optional using the respective notation in ConcurTaskTrees.
- Actions in our task trees become interaction tasks in ConcurTaskTrees without having children.

An example of a task tree for a sequence called *sequence 2082653* generated by our approach and visualized by AutoQUEST is shown in Figure D.1a. *Sequence 2082653* has two children, the first being an iteration of a mouse click, the second being a selection of either an optional text input or a scroll. This visualization is similar to the display of task trees as introduced in this thesis. The same task tree transformed into the ConcurTaskTree notation is shown in Figure D.1b. Also here, *Sequence 2082653* is the root node and it has two children. The children are interconnected with the enabling operator >>, which means that the first child is executed before the second child. The first child is the mouse click,


a) Task tree as generated by our approach:

```
⊟ sequence #2082653
  ⊟ iteration #794214
      LeftMouseClick ⇒ div(id="rev_details_status-selector-wrapper") #764663
  ⊟ selection #2082648
    ⊟ optionality #2082498
      ⊟ iteration #794205
          TextInput ⇒ textarea(id="rev_details_status-comment") #768971
    ⊟ iteration #794179
        Scroll ⇒ body[0] #764614
```

b) Task tree transformed into ConcurTaskTree:

sequence 2082653
LeftMouseClick --> div(id="rev_details_status-selector-wrapper") 764663 *
>>
selection 2082648
[TextInput --> textarea(id="rev_details_status-comment") 768971 *]
[]
Scroll --> body[0] 764614 *

Figure D.1.: Example of a task tree in the notation of our approach (a) and its transformed variant in ConcurTaskTrees visualized using the ConcurTaskTree Environment (b).

which can be iterated. This is indicated through the star at the end of its name. The second child is the selection, which has two children connected with the choice operator []. This means either of the children of the selection is executed. The first child of the selection is the text input, which is iterated (indicated through the star at the end of the name), and which is optional as indicated through the brackets around the name. The second child of the selection is the repeatable scrolling.

E. Additional Plots for Case Study 1

E.1. Sequence Coverage Plots for Case Study 1

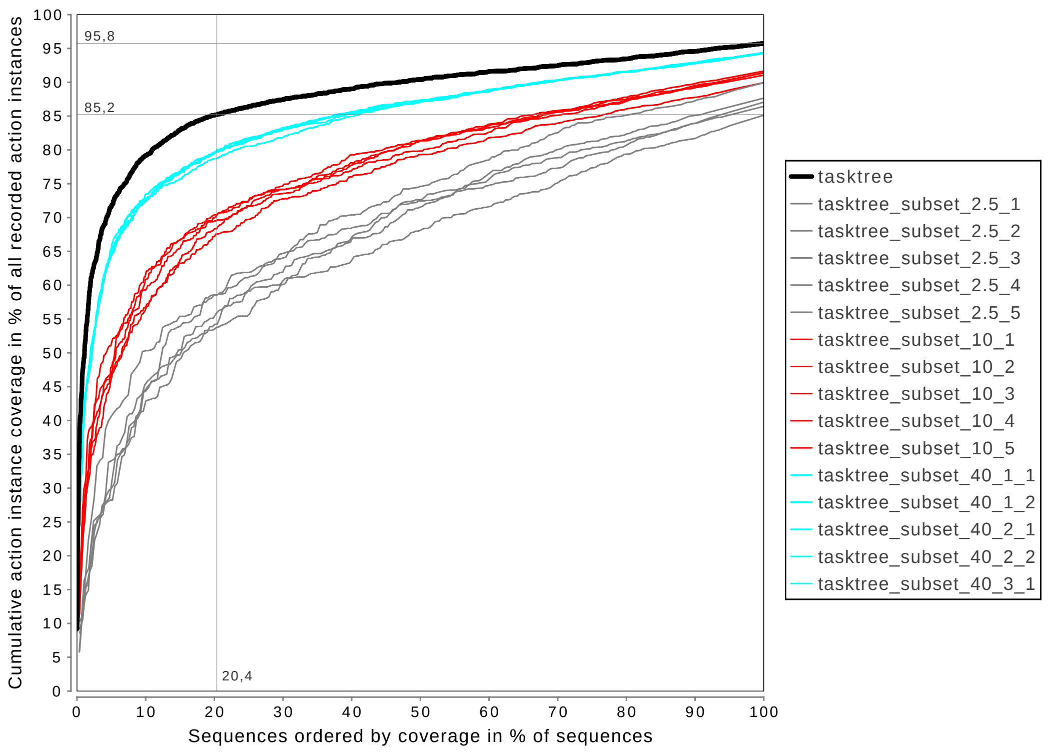


Figure E.1.: Plot for the cumulative action instance coverage of the merged sequences of the reviewer portal data set of the first case study (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

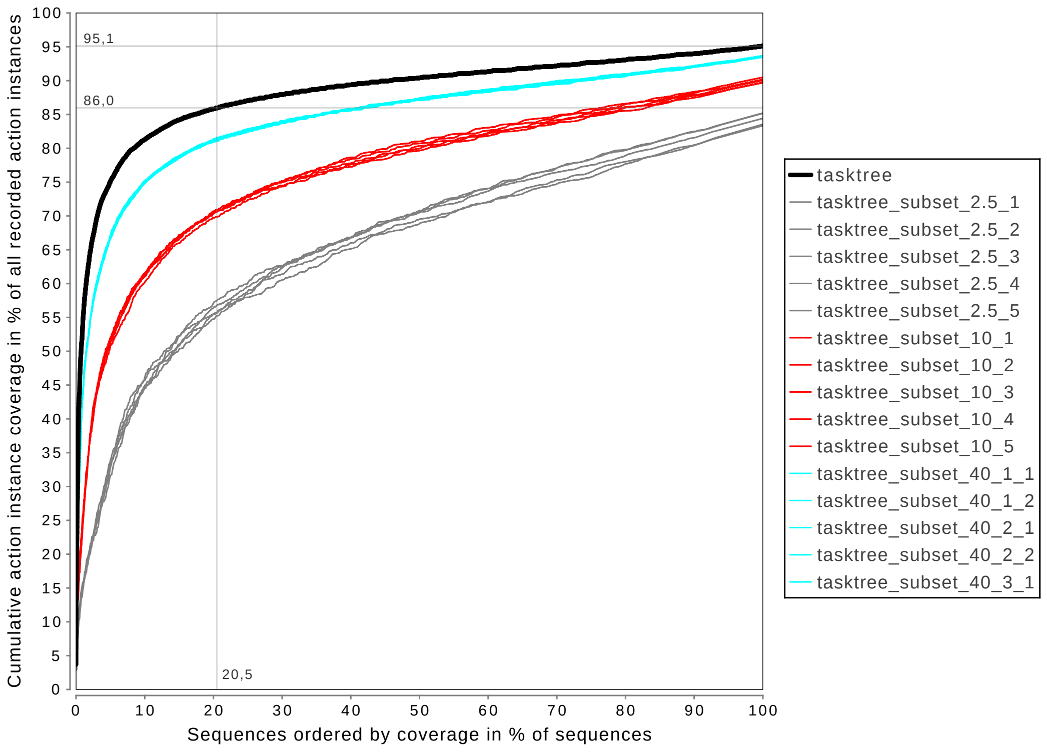


Figure E.2.: Plot for the cumulative action instance coverage of the unmerged sequences of the applicants portal data set of the first case study (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

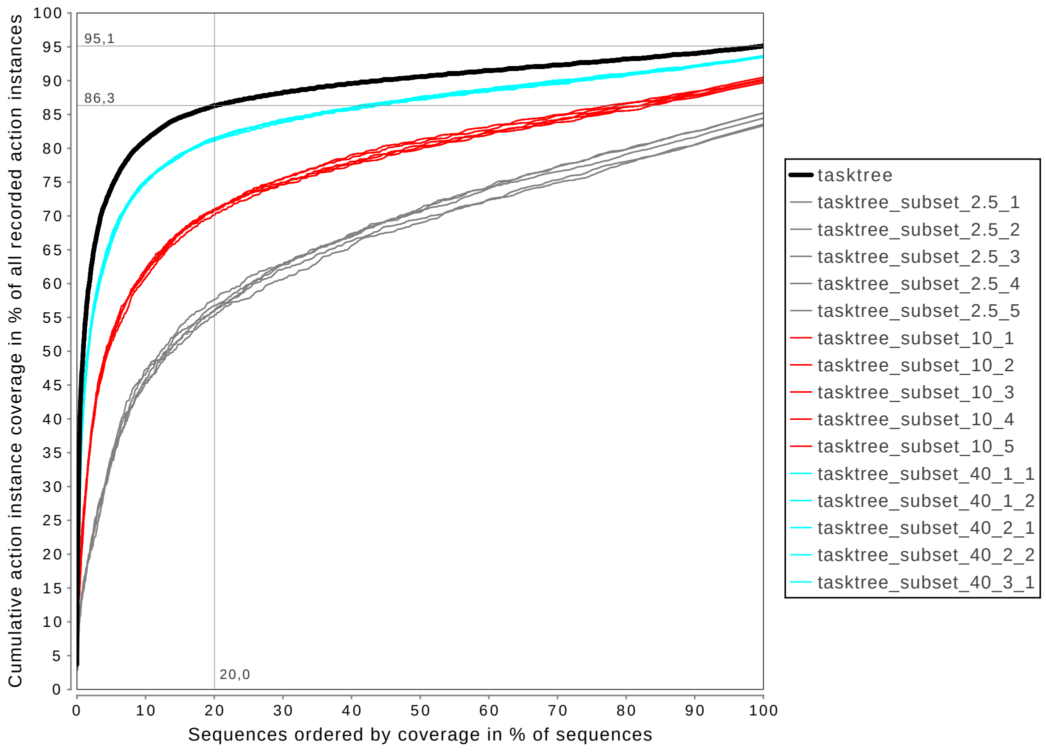


Figure E.3.: Plot for the cumulative action instance coverage of the merged sequences of the applicants portal data set of the first case study (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

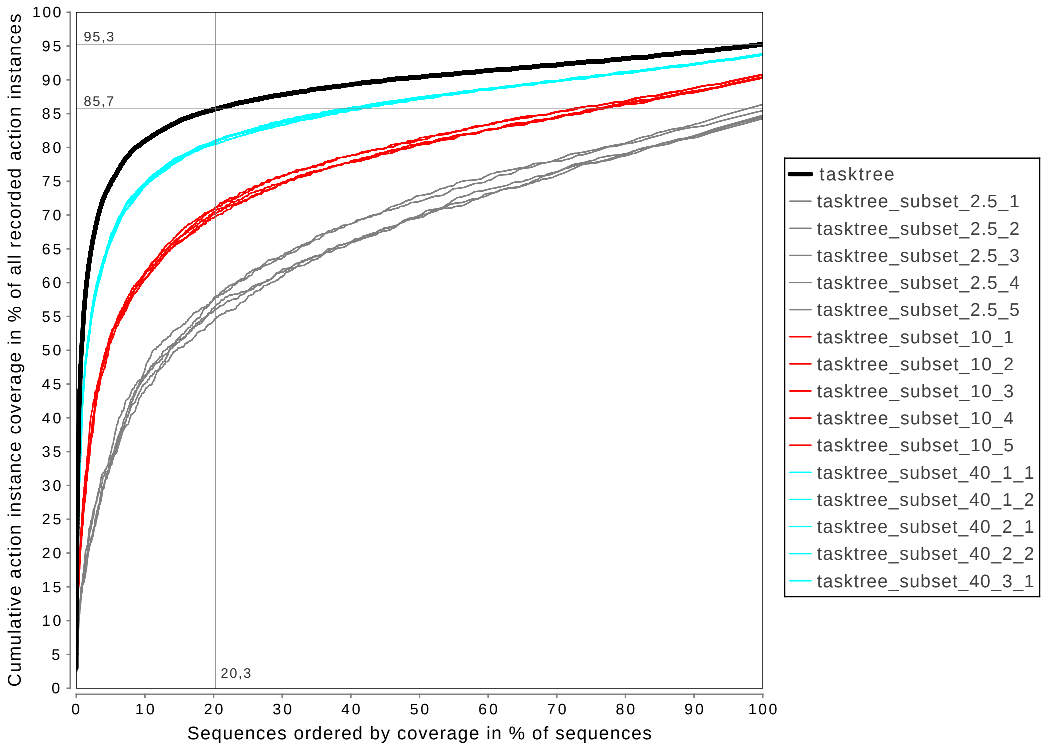


Figure E.4.: Plot for the cumulative action instance coverage of the unmerged sequences of the overall data set of the first case study (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

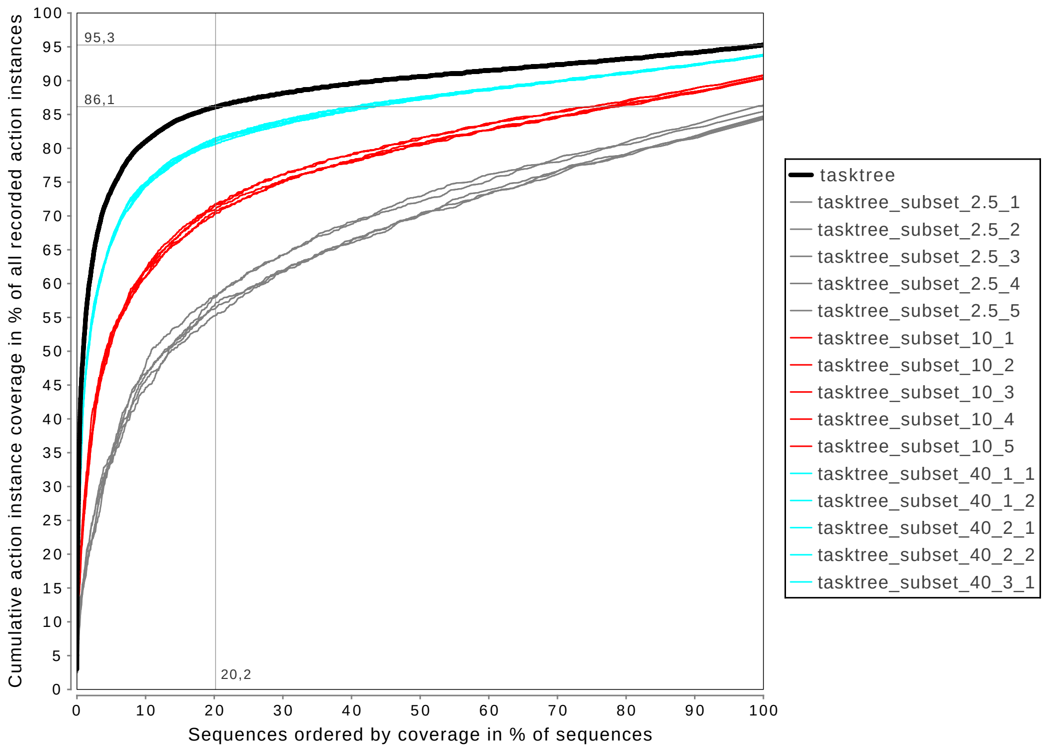


Figure E.5.: Plot for the cumulative action instance coverage of the merged sequences of the overall data set of the first case study (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

E.2. Matches Plots for Case Study 1

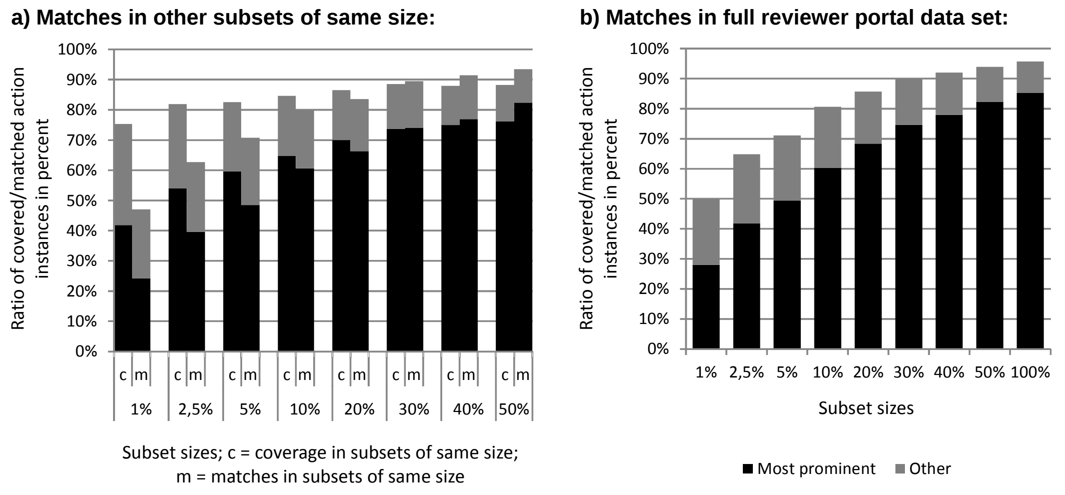


Figure E.6.: Plot for the action instances matched by parsers, which were generated from merged task trees for a specific subset size of the reviewer portal data set in the first case study.

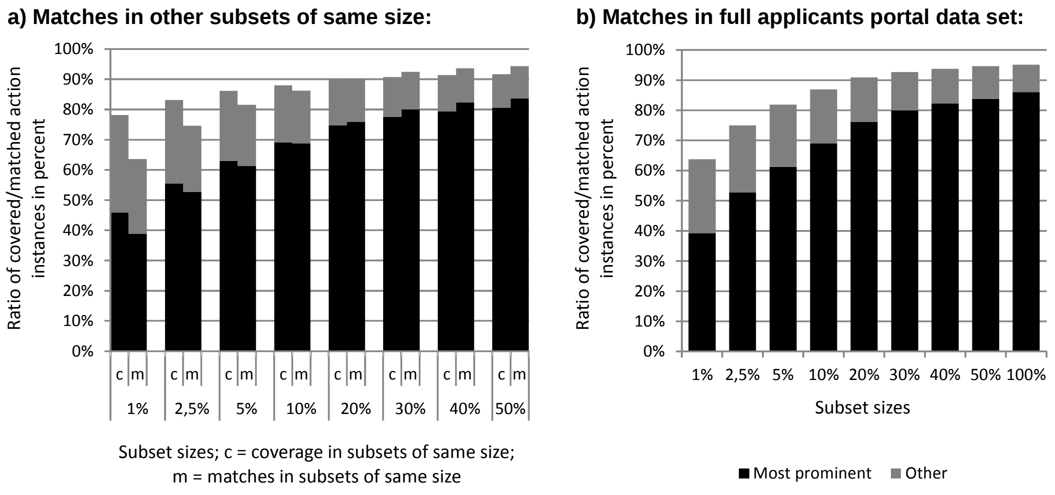


Figure E.7.: Plot for the action instances matched by parsers, which were generated from unmerged task trees for a specific subset size of the applicants portal data set in the first case study.

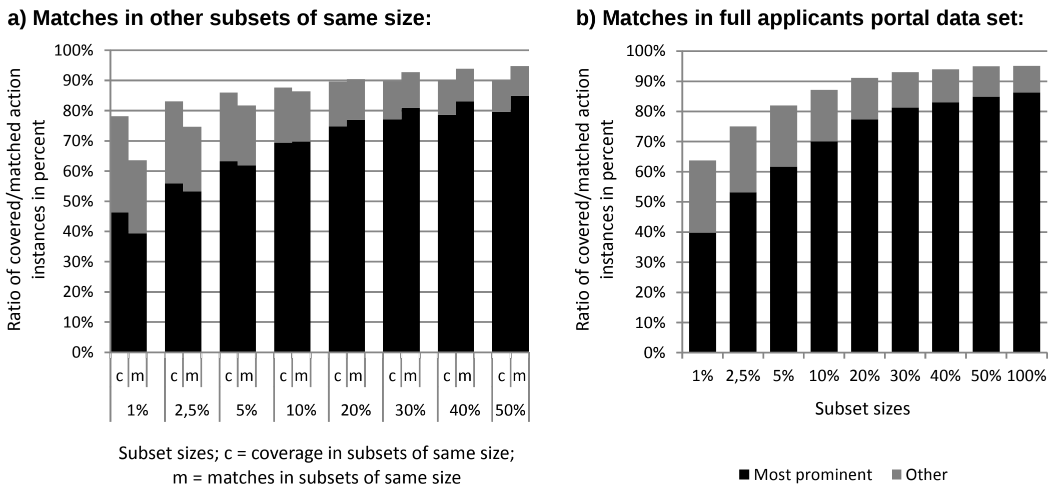


Figure E.8.: Plot for the action instances matched by parsers, which were generated from merged task trees for a specific subset size of the applicants portal data set in the first case study.

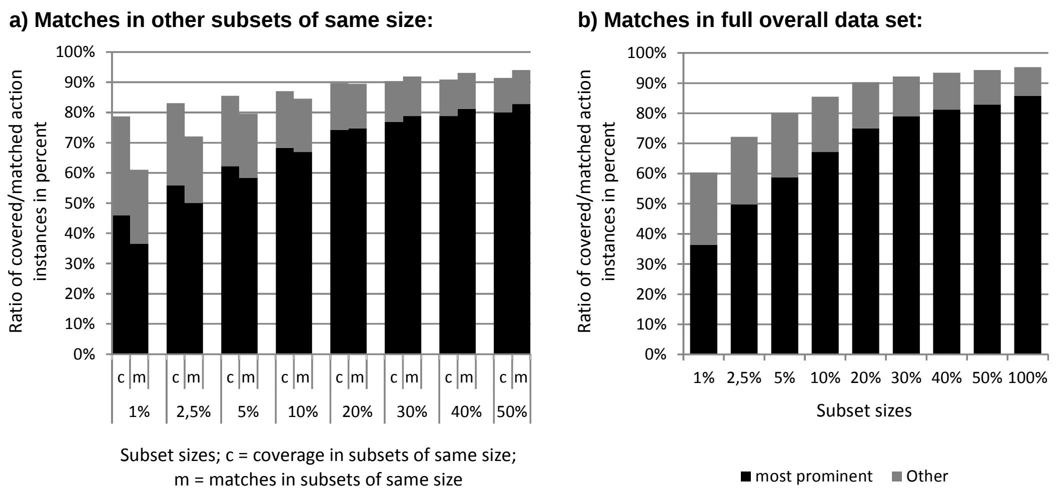


Figure E.9.: Plot for the action instances matched by parsers, which were generated from unmerged task trees for a specific subset size of the overall data set in the first case study.

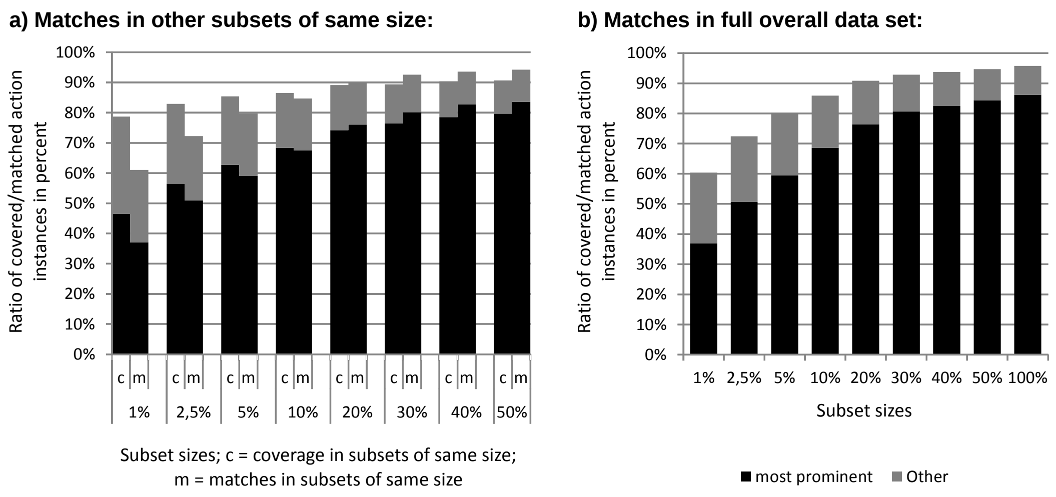


Figure E.10.: Plot for the action instances matched by parsers, which were generated from merged task trees for a specific subset size of the overall data set in the first case study.

F. Additional Plots for Case Study 2

F.1. Sequence Coverage Plot for Case Study 2

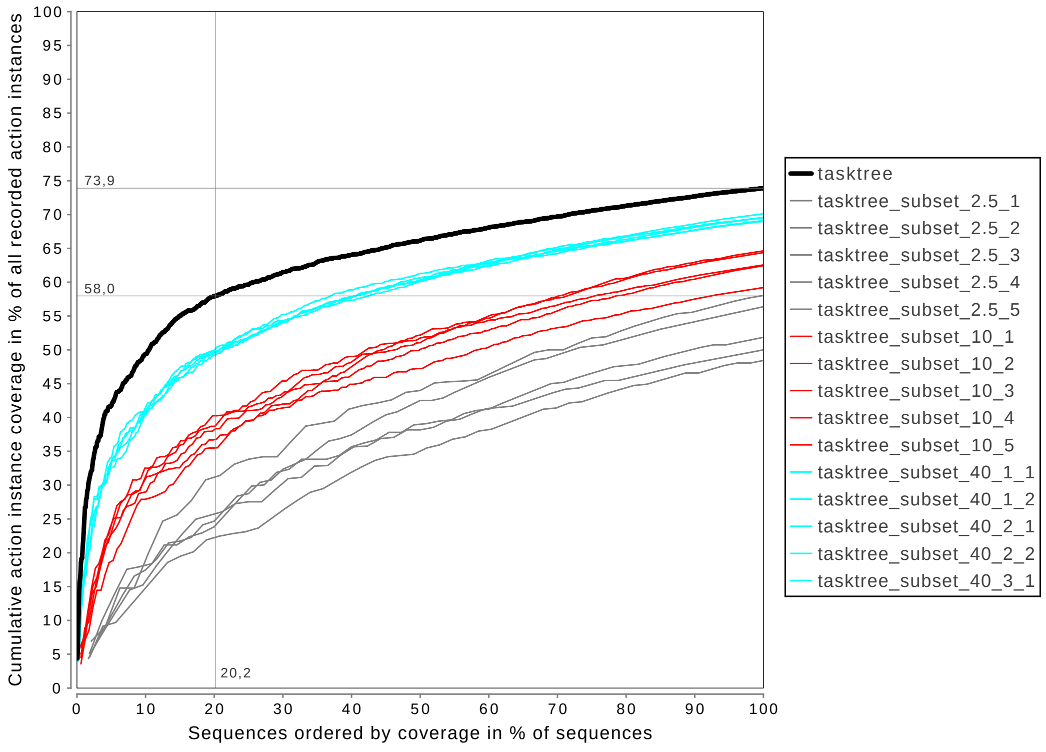


Figure F.1.: Plot for the cumulative action instance coverage of the merged sequences of the new website version (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

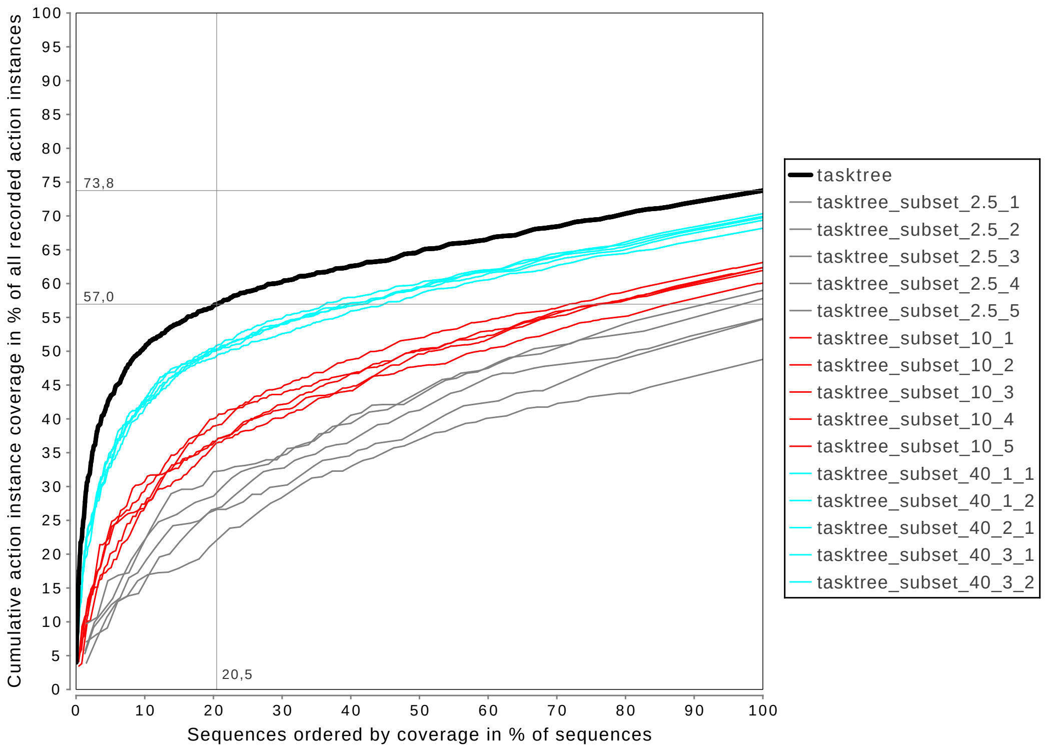


Figure F.2.: Plot for the cumulative action instance coverage of the unmerged sequences of the old website version (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

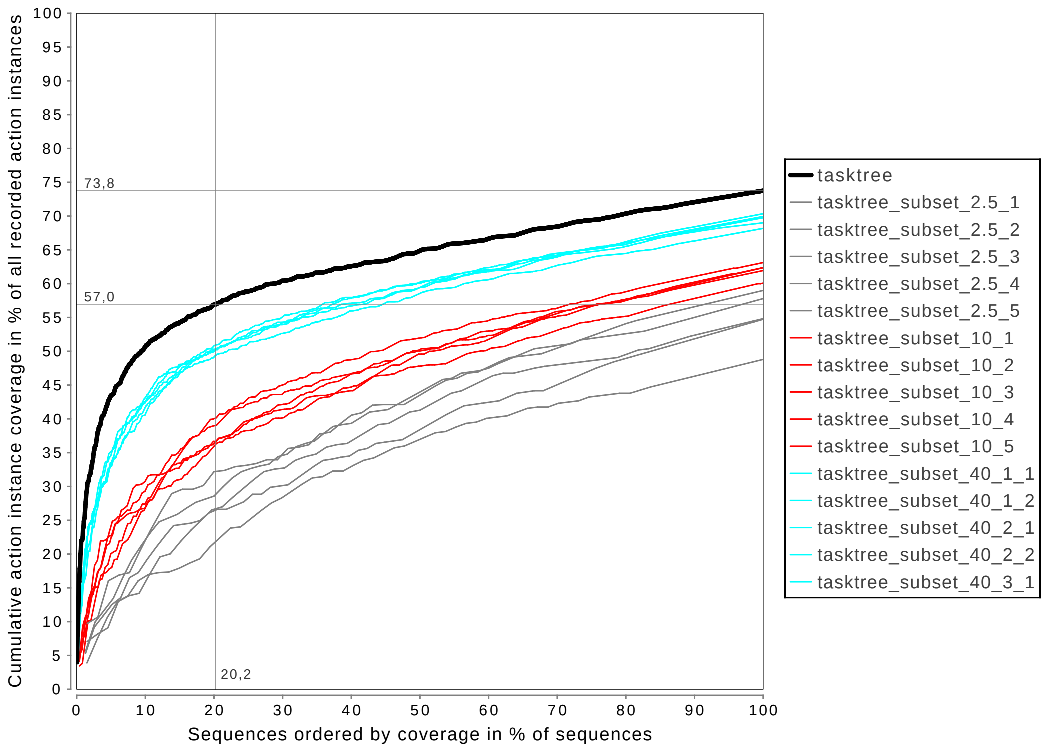


Figure F.3.: Plot for the cumulative action instance coverage of the merged sequences of the old website version (black) and five subsets for the subset sizes 2.5% (grey), 10% (red), and 40% (cyan).

F.2. Matches Plot for Case Study 2

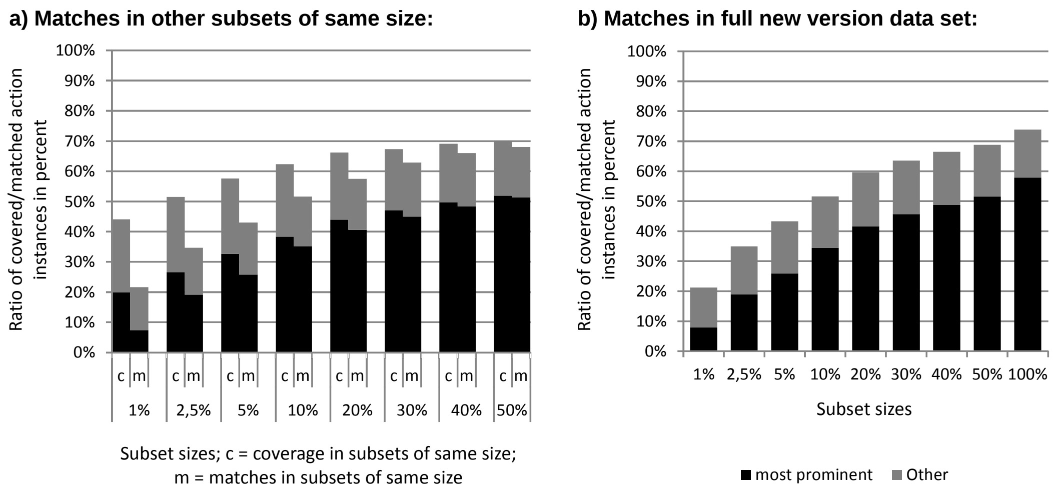


Figure F.4.: Plot for the action instances matched by parsers, which were generated from merged task trees for a specific subset size of the new website version data set in the second case study.

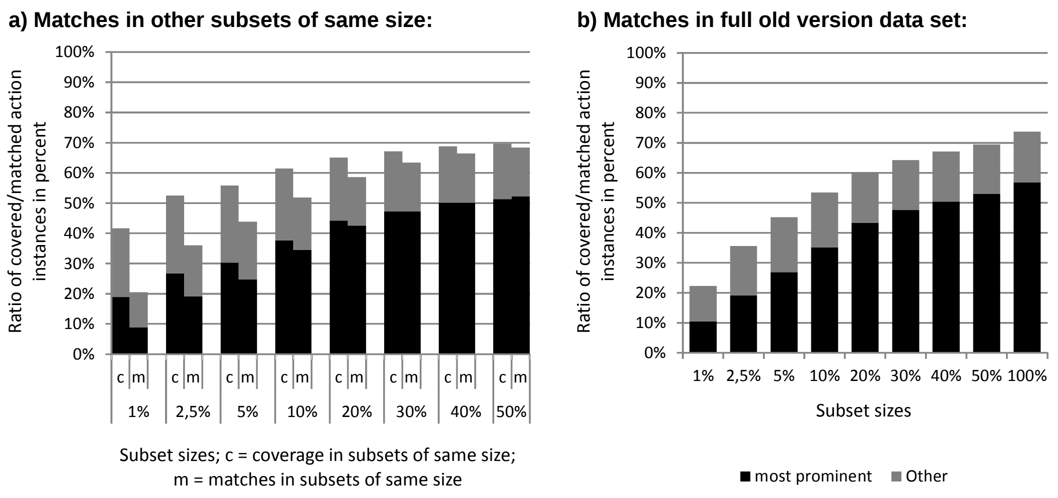


Figure F.5.: Plot for the action instances matched by parsers, which were generated from unmerged task trees for a specific subset size of the old website version data set in the second case study.

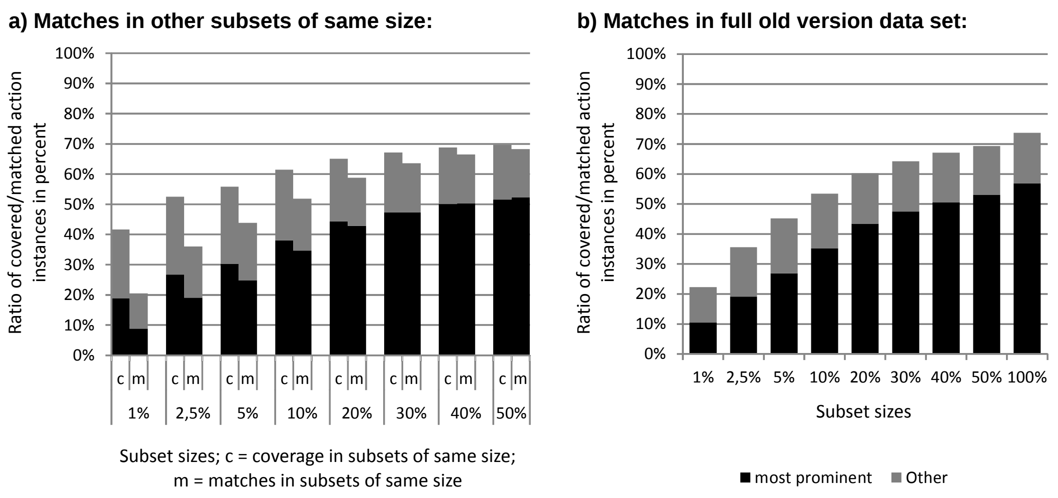


Figure F.6.: Plot for the action instances matched by parsers, which were generated from merged task trees for a specific subset size of the old website version data set in the second case study.