Automatische Anleitung einer Versuchsperson während eines kontrollierten Experiments in ExplorViz

Masterarbeit

Santje Finke

23. September 2014

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Arbeitsgruppe Software Engineering

Betreut durch: Prof. Dr. Wilhelm Hasselbring
M.Sc. Florian Fittkau

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel,

Zusammenfassung

Es existieren verschiedene Tools, mit denen Anwendungen visualisiert werden können, um dem Nutzer Verständnis über die Anwendung zu vermitteln. Um die Vorteile verschiedener Visualisierungsarten zu identifizieren, können kontrollierte Experimente durchgeführt werden. Für diese ist es notwendig, den Probanden eine Einführung in das zu nutzende Tool zu geben und durch Fragen zu testen, wie viel Programmverständnis sie durch das Tool gewinnen. Werden Einführungen von den Versuchsleitern gegeben, kann nicht gewährleistet werden, dass es zu keiner Beeinflussung der Probanden kommt, wodurch Zweifel an der Validität der Ergebnisse entstehen können. Um dies zu verhindern können automatische Tutorials genutzt werden, die in Zusammenarbeit mit den Entwicklern des jeweiligen Tools geplant wurden, sodass eine Erklärung aller Funktionen und ihres Nutzen gewährleistet werden kann. Weitere Vorteile für die Durchführung eines Experiments können aus einem elektronischen Fragebogen gewonnen werden, da dieser sowohl die Zeitmessung als auch die Auswertung erleichtern, sodass diese weniger fehleranfällig ist, als würde sie manuell durchgeführt werden.

In dieser Arbeit werden Konzepte für eine automatische Anleitung einer Versuchsperson während eines kontrollierten Experiments in ExplorViz vorgestellt. Dazu wird ein Überblick über die Auswirkungen verschiedener Anleitungsarten gegeben und erläutert, wie sich ein interaktives Tutorial von diesen absetzt. Außerdem wird beschrieben, was bei der Erstellung eines Fragebogens zu beachten ist, welche Arten von Fragen existieren und welche Skalen als Antwortmöglichkeiten für Multiple-Choice-Fragen vorgegeben werden können. Anschließend werden die Vor- und Nachteile verschiedener Ansätze diskutiert und darauf basierend ein Konzept aufgestellt, anhand dessen das Tutorial und der Fragebogen in ExplorViz integriert werden.

Beide Funktionalitäten kamen in zwei kontrollierten Experimenten zum Einsatz, in dem die Tracevisualisierung von ExplorViz mit der von Extravis verglichen wird. Die Planung, Durchführung und Analyse dieser Experimente wird zum besseren Verständnis ebenfalls in dieser Arbeit erläutert. Basierend auf den während der Experimente gemachten Beobachtungen sowie den Meinungen der Probanden werden das gewählte Konzept und die Umsetzung daraufhin bewertet und mögliche Verbesserungen aufgezeigt.

Sowohl das Tutorial als auch der Fragebogen konnten den Verlauf des Experiments wie geplant unterstützen und wurden von den Probanden gut aufgenommen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Dieses Kapitel beschreibt die Motivation der Arbeit und stellt kurz die gesetzten Ziele vor. Anschließend wird ein Überblick über den Aufbau der Arbeit gegeben.

1.1. Motivation

Um einen Einblick in das Zusammenspiel einzelner Komponenten in komplexen Softwarelandschaften zu bekommen, können unterschiedliche Programme genutzt werden. Je mehr Informationen dargestellt werden können, umso wichtiger wird es, alle Funktionen zu kennen, mit denen die Informationen gezielter ausgewertet werden können, da es andernfalls schwer ist, die gewünschten Informationen zu finden.

ExplorViz [Fittkau u. a. 2013] ist ein solches Programm, das im Zuge dieser Masterarbeit um einen Tutorialmodus und ein Fragebogenkonzept erweitert werden soll.

Der Tutorialmodus soll dazu dienen, den Einstieg in den Umgang mit ExplorViz zu vereinfachen und Einweisungen durch einen Versuchsleiter zu ersetzen. Dadurch wird sicher gestellt, dass die Anleitung nicht von einer oder mehreren Personen abhängt, sondern immer genau gleich abläuft. Gleiche Bedingungen sind insbesondere deswegen, um Experimente mit ExplorViz zu machen, um die Verwendbarkeit zu evaluieren und die Ergebnisse der Experimente durch die Einführung in der Programm nicht beeinflusst werden sollen. Andernfalls können aufgrund von unterschiedlichen Erklärungen verschiedene Ergebnisse auftreten, wodurch die Validität der Experimente anzuzweifeln wäre.

Um die Durchführung der Experimente zu vereinfachen, soll außerdem ein Fragebogenkonzept implementiert werden, sodass die Fragen direkt in die Anwendungsoberfläche integriert werden und nicht in einem separaten Programm oder auf einem Zettel beantwortet werden müssen. Dies vereinfacht sowohl die Auswertung der Ergebnisse, als auch den Ablauf des Experiments für den Teilnehmer, da er sich nur auf die Beantwortung der Frage konzentrieren muss und nicht durch die Fragen von der Aufgabe abgelenkt wird.

1.2. Ziele

Im Folgenden werden die beiden angesprochenen Funktionen, der Tutorialmodus und der Fragebogenmodus, genauer erläutert und beschrieben, welche Funktionalitäten und Eigenschaften sie zur Erfüllung der Vorgaben aufweisen müssen.

Z1.1: Einführung durch Tutorialmodus

Der Tutorialmodus soll derart gestaltet werden, dass schrittweise die Funktionen von ExplorViz erläutert werden. Dazu werden Anweisungen gegeben, welche Elemente angeklickt werden müssen, um ein bestimmtes Verhalten hervorzurufen. Alle anderen Interaktionen werden währenddessen blockiert, um einen festen Ablauf zu gewährleisten, da der Benutzer ansonsten möglicherweise Funktionen nutzt, ohne zu wissen, wie er wieder zur vorherigen Ansicht zurückkehren kann.

Beim erstmaligen Einloggen soll der Tutorialmodus automatisch gestartet werden, bevor der Benutzer das Programm im normalen Modus nutzen kann. Wurde das Tutorial beendet, soll es auf Wunsch erneut gestartet und zu einem beliebigen Zeitpunkt abgebrochen werden können.

Damit das Tutorial immer unter den gleichen Bedingungen durchgeführt werden kann, sollte es nicht auf aktuellen Live-Daten sondern auf einem definierten System ausgeführt werden. Dieses muss entsprechende Eigenschaften aufweisen, um alle Funktionen sinnvoll erklären zu können. Insbesondere ist daran zu denken, dass auch die Zeitleistenfunktion genutzt wird. Dazu muss im Verlauf des Tutorials ein neuer Zustand geladen werden, um das Verhalten der Live-Überwachung zu simulieren.

Optionales

Eine zusätzliche Funktion des Tutorialmodus wäre die Unterstützung verschiedener Sprachen. Welche Sprache für das Tutorial genutzt wird, sollte dazu auf dem Server festgelegt werden können, anstatt durch den Benutzer ausgewählt zu werden. Dadurch wird sicher gestellt, dass die Teilnehmer die gleichen Anweisungen bekommen und das Ergebnis des Experiments nicht durch eine schlechte Übersetzung verzerrt wird.

Z1.2: Lenkung durch Fragebogenkonzept

Das Fragebogenkonzept soll bei Bedarf automatisch nach Beendigung des Tutorialmodus gestartet werden. Daraufhin sollen Daten wie Alter, Geschlecht und Erfahrungsgrad des Benutzers abgefragt und der Fragebogen gestartet werden. Im Anschluss an die Aufgaben soll qualitatives Feedback vom Benutzer erfragt werden.

Nach Hauptteil des Fragebogens soll dazu dienen, Experimente mit ExplorViz durchzuführen. Sowohl Multiple Choice- als auch Freitext-Fragen sollen möglich sein. Außerdem muss berücksichtigt werden, dass bei einer Live-Visualisierung bestimmte Fragen nach fortgeschrittener Zeit schwer bis unmöglich zu beantworten sind. Um dieses Problem zu umgehen, wurde von der Nutzung der Live-Visualisierung abgesehen. Stattdessen muss das zu visualisierende System im Voraus überwacht werden, sodass diese Aufzeichnungen anstelle von Live-Daten angezeigt werden können. Anhand dieser Aufzeichnungen soll es dann auch möglich sein, Fragen auf bestimmte Zeiträume zu begrenzen, indem die Visualisierung angehalten wird, bis bestimmte Fragen beantwortet worden sind.

Zu jeder Frage soll neben den gegebenen Antworten und der Benutzerkennung auch ein Zeitstempel gespeichert werden, der angibt, wann die Frage beantwortet wurde, so dass hinterher bestimmt werden kann, wie viel Zeit der Nutzer zur Beantwortung der Frage benötigt hat. Hierdurch wird auch eine Zuordnung zum User Trace [Kosche 2013] möglich, sodass nachvollzogen werden kann, welche Aktionen für die Beantwortung welcher Frage ausgeführt wurden. Die Ergebnisse sollen im CSV Format gespeichert werden und exportieren werden können.

Optionales

Zusätzlich wäre ein Menüpunkt denkbar, mit dem die Ergebnisse auch sofort ausgewertet werden können. Diese Funktion wäre Administratoren vorbehalten. Eine weitere Funktion für das Administrationsmenü wäre auch die Möglichkeit, die Fragen über eine Oberfläche erstellen zu können, anstatt sie direkt im jeweiligen Format auf dem Server abzulegen.

Z2: Beteiligung an kontrollierten Experimenten zum Vergleich von Tracevisualisierungen

Ein weiterer Bestandteil dieser Arbeit ist die Beteiligung an der Planung und Durchführung von zwei kontrollierten Experimenten, in dem der Tutorialmodus und der Fragebogen zum Einsatz kommen sollen. In den Experimenten soll die von ExplorViz genutzte Art der Visualisierung mit einer anderen verglichen werden und herausgefunden werden, ob verschiedene Visualisierungen unterschiedlich gut zum Verständnis beitragen. Dazu müssen ein Vergleichstool, ein zu analysierendes Programm ausgesucht sowie Aufgaben aufgestellt werden und anschließend die Ergebnisse des Experiments ausgewertet werden.

1.3. Aufbau

Zunächst werden in Kapitel 2 die Technologien beschrieben und erläutert, die zur Umsetzung der Ziele genutzt werden. Anschließend werden in Kapitel 3 und Kapitel 4 verschiedene Einweisungsarten und Fragebögen diskutiert, um ein besseres Verständnis für die Thematik zu erhalten. Darauf aufbauend folgt in Kapitel 5 der Ansatz und die Planung der Implementierung, welche anschließend in Kapitel 6 beschrieben wird.

Die Experimente, welche die implementierten Funktionen nutzen, werden in Kapitel 7 beschrieben und die daraus gewonnenen Ergebnisse für das Tutorial und den Fragebogen werden in Kapitel 8 dargestellt. In Kapitel 9 werden verwandte Arbeiten vorgestellt und in Kapitel 10 erfolgt ein Ausblick sowie eine Zusammenfassung der wichtigsten Punkte der Arbeit.

Kapitel 2

Grundlagen und Technologien

In diesem Kapiel wird das zu erweiternde Programm ExplroViz vorgestellt sowie die genutzten Sprachen und Frameworks erläutert. Außerdem wird der Begriff des kontrollierten Experiments beschrieben, da dieses die Motivation für Tutorial und integrierten Fragebogen ist.

2.1. ExplorViz

ExplorViz ist das Visualisierungstool, dessen Erweiterung um eine automatische Anleitung Ziel der Arbeit sein wird. Um das Verständnis für und das Wissen über eine Softwarelandschaft zu fördern, werden Verhalten und Kommunikation der Systeme untereinander durch Echtzeitvisualisierung dargestellt. Die Visualisierung basiert auf Traces, die von Monitoring-Tools erstellt und an ExplorViz weiter gegeben werden. Um die darin enthaltenen Informationen darzustellen, existieren zwei verschiedene Visualisierungsebenen.

Die Landschaftsdarstellung (siehe Abbildung 2.1), die Ähnlichkeiten zu einem UML-Diagramm aufweist, gibt einen groben Überblick über das gesamte Softwaresystem. Die Anwendungen der einzelnen Systeme werden in Knotengruppen unterteilt und die Kommunikation zwischen den Anwendungen wird dargestellt.

In der Anwendungsebene (siehe Abbildung 2.1) wird eine der Anwendungen angezeigt und in 3D anhand der Stadtmetapher (siehe [Wettel und Lanza 2007]) visualisiert. Die Gebäude der Stadt entsprechen dabei den Komponenten und Klassen, wobei die Komponenten geöffnet werden können, um die darin befindlichen Komponenten und Klassen anzuzeigen.

Des Weiteren verfügt ExplorViz über einen Time-Shift-Modus, mit dem die Echtzeitvisualisierung verlassen werden kann, um zu früheren Zeitpunkten der Visualisierung zurückzukehren, so dass Vergleiche von unterschiedlichen Zeitpunkten leichter möglich sind, als wenn nur das aktuelle Verhalten sichtbar wäre [Fittkau u. a. 2013].

2.2. Xtend

Xtend ist eine Programmiersprache, die in Javacode übersetzt wird und dadurch mit Java kompatibel genutzt werden kann. Sie bietet viele Erweiterungen zu Java, durch die das

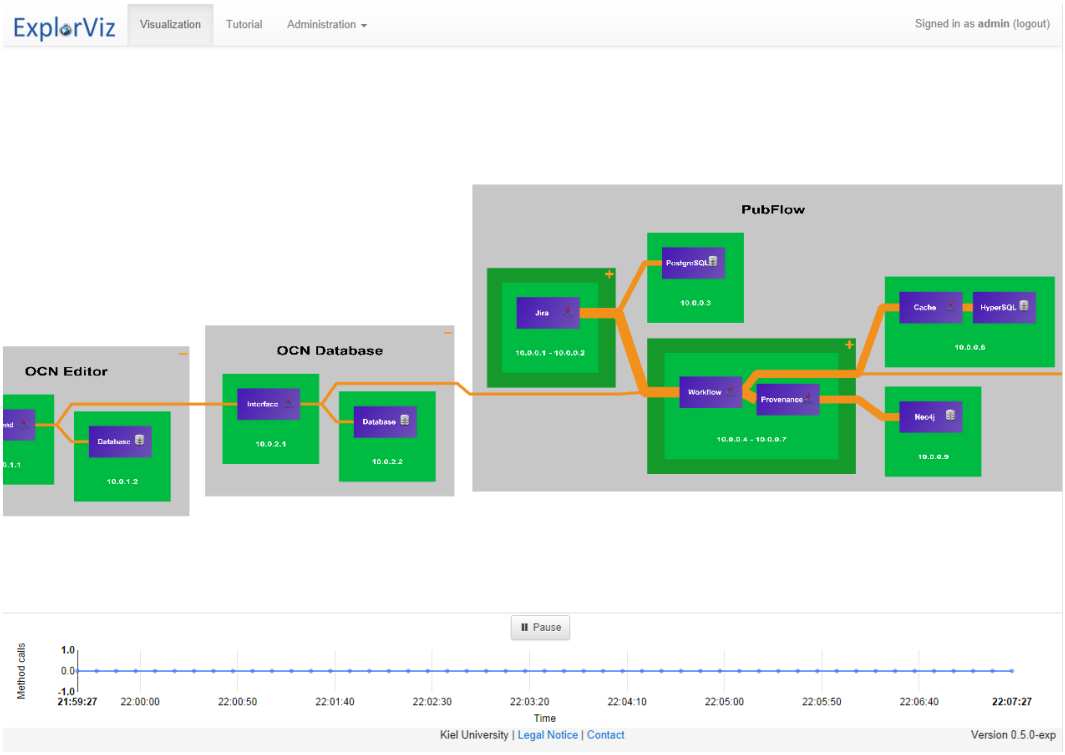

Abbildung 2.1. Screenshot der Landschaftsansicht von ExplorViz

Programmieren vereinfacht werden soll, wie zum Beispiel Lambda-Ausdrücke, Typinferenze und Properties. Listing 2.1 zeigt einige Unterschiede anhand der Step-Klasse auf, die ein wichtiger Bestandteil des Tutorials (Abschnitt 6.1) ist. Zudem werden alle Anweisungen als Ausdrücke behandelt und können dadurch auf der rechten Seite von Zuweisungen genutzt werden. Der Rückgabewert ergibt sich dabei aus dem letzten Ausdruck [*Xtend User Guide*]. Durch die Übersetzung in Java-Code ist Xtend problemlos zusammen mit GWT nutzbar [*Xtend User Guide*].

Xtend wurde auf Basis von Xbase entwickelt, einer Domain Specific Language (DSL), die wiederum mit dem Xtext Framework entwickelt wurde und in jeder Xtext basierten Sprache genutzt werden kann [Efftinge u. a. 2012].

Listing 2.1. Beispiel einiger Unterschiede zwischen Xtend und Java

```
class Step implements IsSerializable {
  @Property String source = ""
  // @Property generiert automatisch Getter und Setter
  @Property boolean doubleClick = false
```

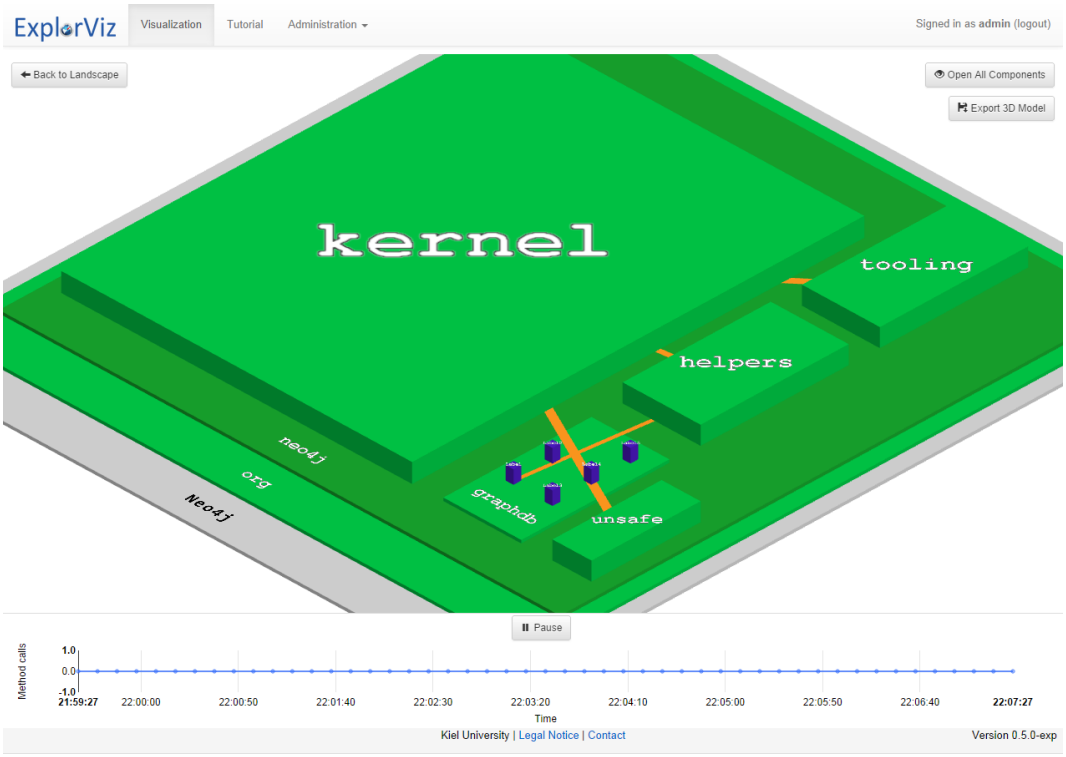

Abbildung 2.2. Screenshot der Anwendungsansicht von ExplorViz

```
  // Attribute ohne Sichtbarkeitsschlüsselwort sind private
  @Property boolean rightClick = false
  // Zeilen müssen nicht mit einem Semikolon abgeschlossen werden
  @Property boolean leftClick = false
  @Property boolean hover = false

  // Konstruktoren werden nur mit dem Schlüsselwort new ausgezeichnet
  new(String source, boolean doubleClick, boolean rightClick,
    boolean leftClick, boolean hover){
    this.source = source
    this.doubleClick = doubleClick
    this.rightClick = rightClick
    this.leftClick = leftClick
    this.hover = hover
  }
}
```

```
// Attribute mit Getter können abgefragt werden, als wären sie
// öffentlich sichtbar
def static incTutorial(String source, String target, boolean left,
  boolean right, boolean hover) {
  if (tutorial) {
    val step = getStep()
    // step.connection, step.source etc. werden in step.isConnection und
    // step.getSource übersetzt, abhängig vom Rückgabetyp
    if (step.connection && source.equals(step.source)
      && target.equals(step.dest) &&((left && step.leftClick) ||
      (right && step.rightClick) || (hover && step.hover))) {
      incStep()
    }
  }
}
```

2.3. Google Web Toolkit (GWT)

GWT ist ein Framework zur Entwicklung von Webseiten und stellt Java APIs zur Verfügung, mit denen Anwendungen in Java programmiert und nach JavaScript kompiliert werden können. Dadurch wird es möglich, JavaScript-basierte Anwendungen mit nur geringen Kenntnissen von JavaScript zu entwickeln. Nur wenn es nicht möglich ist, eine Funktionalität in Java zu implementieren ist es nötig, Funktionen direkt in JavaScript zu schreiben und diese in den Java-Code einzubinden [*GWT*].

Unter anderem wird auch ein RPC-Framework bereitgestellt, das die Übertragung von Java-Objekten zwischen Server und Client vereinfacht. GWT RPC basiert auf der Java Servlet Architektur und generiert automatisch eine Proxy-Klasse, um mit dem Server zu kommunizieren [*GWT RPC-Tutorial*]. Ein Beispiel ist in Listing 2.2 zu sehen.

Listing 2.2. Verwendung von GWT RPC

```
package explorviz.server.experiment;
public class TutorialServiceImpl extends RemoteServiceServlet implements
TutorialService {
  @Override
    public String getText(final int number){
      \\ Einlesen des durch die Nummer bezeichneten Tutorialtextes
    }
}
package explorviz.visualization.experiment.services;
@RemoteServiceRelativePath("tutorial")
public interface TutorialService extends RemoteService {
```

```
  public String getText(int number);
}
public interface TutorialServiceAsync {
  void getText(int number, AsyncCallback<String> callback);
}

//Aufruf in xtend-Datei
val TutorialServiceAsync tutorialService = GWT::create(typeof(TutorialService))
tutorialService.getText(number, new TextCallback())

class TextCallback() implements AsyncCallback<String>{
  override onFailure(Throwable caught){
  }
  override onSuccess(String result){
    //do something with the result
  }
}
```

2.4. JavaScript

Da ExplorViz verschiedene externe JavaScript-Libraries nutzt, wird trotz GWT an vielen Stellen natives Javascript benötigt, um die Interaktion mit dem Benutzer zu erleichtern, da für diese Libraries kein Javainterface vorhanden ist. Insbesondere werden dabei drei Libraries benutzt, die im Folgenden näher vorgestellt werden.

2.4.1. JQuery und JQuery UI

JQuery [*JQuery*] ist eine viel genutzte JavaScript Library. Sie stellt leicht zu nutzende Funktionen zum Durchsuchen und Manipulieren von HTML-Dokumenten zur Verfügung. Häufig benötigte Suchmuster und Manipulationsarten sind vorgefertigt integriert und ohne weitere Kenntnis nutzbar, während speziellere Anfragen durch Verschachtelung von Funktionen selbst zusammengestellt werden müssen. Außerdem werden EventHandling, Animationen und AJAX-Nutzung durch zusätzliche, über Optionen anpassbare Funktionen unterstützt und erleichtert.

JQuery UI [*JQuery UI*] ist eine auf JQuery aufgesetzt Library, die auf Benutzeroberflächen spezialisiert ist. Sie stellt verschiedene Oberflächenelemente zur Verfügung, die durch viele Optionen an die Wünsche des Entwicklers angepasst werden können. Zusätzlich werden mehrere CSS-Themes angeboten, sodass auch Nutzer mit geringen CSS-Kenntnissen nicht auf das Basis-Design beschränkt sind.

2.4.2. Bootstrap

Bei Bootstrap [*Bootstrap*] handelt es sich um eine JavaScript Library, die verschiedene Optionen zur Verfügung stellt, um Eingabeformulare ohne viel Aufwand ansprechend und zweckdienlich zu formatieren. Dabei ist es nicht nötig, Unterscheidungen anhand des genutzten Anzeigemediums zu machen, da die Skalierung für verschiedene Medien von den jeweiligen Bootstrap-Komponenten übernommen wird. Um die Bootstrap-Eigenschaften für ein Formular zu erhalten, genügt es in den meisten Fällen, die entsprechende Bootstrap-Klasse in das HTML-Element zu schreiben. Dadurch werden Gruppierungen, Anpassungen an die Seitenverhältnisse und Größenanpassungen der Elemente innerhalb eines Formulars automatisch vorgenommen.

Zusätzlich werden Glyphs, kleine Icons, bereitgestellt, die für sprachunabhängige Beschriftung von Buttons oder als Addons an Inputs genutzt werden können.

2.5. Kontrolliertes Experiment

Bei einem kontrollierten Experiment werden die Testpersonen in zwei Gruppen unterteilt, die Experimentalgruppe und die Kontrollgruppe. Die Experimentalgruppe ist die Gruppe, anhand derer die Ergebnisse gewonnen werden sollen, während die Kontrollgruppe dazu dient, die Ergebnisse zu validieren und Zufall oder äußere Einwirkungen auf das Ergebnis auszuschließen. Dabei ist es wichtig, dass Unterschiede zwischen den beiden Gruppen nur durch das Experiment selbst zustanden kommen, sie aber ansonsten den gleichen Bedinungen ausgesetzt sind und keine Unterschiede zwischen ihnen bestehen, die sich auf das Experiment auswirken. Erfolgt die Zuordnung der Testpersonen zu den beiden Gruppen zufällig, wird auch von einem Zufallsexperiment gesprochen.

Selbst wenn die Zuteilung zufällig vorgenommen wird und die Gruppen sich auf den ersten Blick nicht stärker voneinander unterscheiden, als es unvermeidbar ist, können die aus einem Experiment folgenden Schlussfolgerungen aus unterschiedlichen Gründen angezweifelt werden. Diese Gründe werden in vier verschiedene Kategorien eingeteilt: Statistical Conclusion Validity (Validität der statistischen Schlussfolgerug), Internal Validity, Construct Validity (Validität der Konstruktion) und External Validity [Shadish u. a. 2002]. Ziel der zu entwickelnden Experimentumgebung für ExplorViz ist es, die interne Validität zu erhöhen, die im Folgenden beschrieben ist. Weitere Informationen auch über die anderen Arten von Validität können [Shadish u. a. 2002] entnommen werden.

2.5.1. Interne Validität

Bei der Frage nach der internen Validität geht es darum, ob das Ergebnis eines Experiments tatsächlich dem Experiment zuzuschreiben ist und nicht durch andere Umstände hervorgerufen wurde. Aufgabe des Forscher ist es daher, andere mögliche Erklärungen für seine Ergebnisse auszuschließen. Ansatzpunkte dafür können die Auswahl der Personen,

Beeinflussung durch andere Ereignisse, Nichtbeenden des Experiments, mehrmalige Experimente mit den selben Personen oder natürliche, zeitlich bedingte Veränderungen sein [Shadish u. a. 2002].

Auswirkungen durch die Auswahl der Testpersonen und die natürliche Entwicklung einer Person können durch die zufällige Auswahl zwar nicht verhindert, aber ausreichend eingeschränkt werden. Schwieriger ist es, den Einfluss von Ereignissen auszuschließen, die im Zusammenhang mit dem Experiment vorkommen und nicht unterbunden werden können [Shadish u. a. 2002]. Im Fall des ExplorViz-Experiments wäre ein solches Ereignis die Einleitung durch den Experimentleiter. Damit die Ergebnisse nicht auf die gegebene Einleitung zurückgeführt werden können, muss sie für jede Testperson auf die gleiche Weise erfolgen, weshalb diese Aufgabe automatisch ausgeführt werden soll, um Manipulationen zu verhindern.

Kapitel 3

Einweisungen

Im folgenden Kapitel werden die Unterschiede und Auswirkungen von verschiedenen Methoden vorgestellt, die zur Einweisung eines Benutzers in eine ihm unbekannte Thematik genutzt werden können. Unterschieden wird dabei zwischen umfassenden Anleitungen und mit Beispielen arbeitenden Tutorials.

3.1. Anleitung

Charney und Reder [1986] nach lassen sich Anleitungen, das heißt Texte, die dazu dienen, dem Leser die Handhabung eines Gegenstandes oder Programms zu vermitteln, in verschiedene Kategorien einteilen. Auf der einen Seite werden Beispiele genutzt, anhand derer die Erklärungen durchgeführt werden und dem Leser dadurch einen detaillierten Vorgehensplan geben. Auf der anderen Seite gibt es Anleitungen, die dem Leser nur das Nötigste mitteilen, nicht alle Möglichkeiten vorstellen und ihn stattdessen dazu auffordern, sich selbst mit dem Problem auseinander zu setzen und sich mit den gegebenen Mitteln vertraut zu machen.

Besonders Computerbenutzern wird nachgesagt, sich ungern mit längeren Texten auseinander zu setzen und stattdessen lieber selbst herauszufinden, wie ein Programm funktioniert, weshalb Tutorials und das angeleitete Erkunden sich für solche Anwendungen anbieten. Im Gegensatz zu den bloßen Erklärungen an Beispielen, muss der Benutzter sich bei der Erkundung selbstständig Ziele setzen, die er erreichen will und herausfinden, wie er dieses Ziel erreichen kann. Problematisch ist dabei, dass der Benutzer sich dabei sehr wahrscheinlich mit der ersten gefundenen Lösung zufrieden geben wird, selbst wenn es eine bessere gäbe, die er einer Anleitung am Beispiel hätte entnehmen können. Ebenso kann es vorkommen, dass der Benutzer bestimmte Funktionen zur Erfüllung seiner Ziele nicht benötigt und sich daher nicht mit ihnen vertraut macht, da das Wissen um die Möglichkeiten, die die Software bietet, nicht bekannt ist, was durch Beispiele verhindert werden kann [Charney und Reder 1986].

Festgestellt wurde, dass Benutzer, die eine rein theoretische Anleitung bekommen haben, gute Ergebnisse bei der Anwendung direkt nach der Einarbeitung erzielen. Im Gegensatz dazu liefern die Benutzer, die das Programm selbstständig erkundet haben, nach einigen Tagen deutlich bessere Ergebnisse, da die praktische Anwendung das Wissen gefestigt hat. Daraus lässt sich schließen, dass die Erinnerungen an Beispiele flüchtiger

sind als Erinnerungen an selbst erarbeitetes Wissen [Charney und Reder 1986].

Eine weitere Unterscheidung, die Charney u. a. [1988] vornehmen, besteht in der Ausführlichkeit der gegebenen Informationen. Dabei beschäftigen sie sich mit der Frage, ob zusätzliche, nicht essentielle Informationen eher hinderlich sind, und von den wichtigen Punkten ablenken, oder durch die Ergänzungen für ein besseres Verständnis sorgen. Dazu wird in drei Kategorien unterschieden, welches Wissen vermittelt werden soll: Welche Bedeutung und welchen Nutzen haben die zu lernenden Funktionen, wie werden sie korrekt ausgeführt und wann sollten sie ausgeführt werden. Dabei wurde festgestellt, dass Ausführlichkeit sich nur bei der Erläuterung der Anwendung positiv auswirkt, während längere Erklärungen der anderen beiden Punkte vernachlässigbare Auswirkungen haben. Weiterhin wurde auch bemerkt, dass ausführliche Anleitungen kaum Einfluss haben, wenn der Leser schon weiß, welche Aufgabe er hinterher ausführen muss, während Leser, die ihre Aufgabe noch nicht kennen, stärker von ausführlicheren Anweisungen profitieren. Das liegt daran, dass bei einer bekannten Aufgabe zielgerichtet gelesen werden kann und nur das Wissen aufgenommen wird, das als hilfreich erachtet wird, während bei einer unbekannten Aufgabe ein besseres Grundverständnis benötigt wird [Charney u. a. 1988].

3.2. Tutorial

Tutorials geben Anweisungen anhand von Beispielen und vermitteln dadurch ganz konkrete Lösungswege für bestimmte Probleme, weshalb sie sich besonders dann eignen, wenn die Anwendung direkt nach dem Tutorial erfolgt. Der Lösungsweg wird Schritt für Schritt erklärt, sodass der Benutzer sie nachvollziehen und das Beispiel entsprechend der Anweisungen bearbeiten kann. Der Nachteil dieser genauen Anweisungen liegt dabei darin, dass es leicht fällt, die Anweisungen mechanisch zu befolgen anstatt darüber nachzudenken, warum genau diese Schritte ausgeführt werden sollen. Im Falle von Computerprogrammen enthalten Tutorials oft neben den Anweisungen auch Informationen darüber, welche Rückmeldungen vom Programm zu erwarten sind [Charney und Reder 1986].

Da ein gut strukturiertes Tutorial starke Ähnlichkeit mit einer Einweisung hat, die durch eine Person gegeben würde, eignen sie sich insbesondere für Anwendungen, die gut durch mündliche Anleitungen und Erklärungen erläutert werden können. Eine solche mündliche Wissensweitergabe erfordert jedoch personelle Mittel und kann nur vor Ort gegeben werden. Ein Tutorial hingegen kann von einer beliebigen Anzahl Personen zur gleichen Zeit an dem Ort und zu der Zeit genutzt werden, zu der der Benutzer die Hilfe benötigt und kann ebenso leicht wiederholt werden, wenn Unklarheiten entstehen. Wie genau dabei auf besondere, mögliche Fragen eines Benutzers eingegangen wird, hängt von dem Tutorial ab, aber alle möglichen Fragen und Probleme zu berücksichtigen ist zumeist nur sehr schwer möglich [Nixon u. a. 2009].

Es lassen sich zwei Arten von Tutorials unterscheiden, die im Folgenden näher erläutert werden.

3.2.1. Textuelles Tutorial

Bei rein textuellen Anweisungen, sowohl geschrieben als auch gesprochen, liegt es in der Verantwortung des Benutzers, den Anweisungen zu folgen und nicht in Situationen zu geraten, aus denen er den Ausweg nicht kennt. Genauso müssen Fehler selbstständig korrigiert werden, falls diese einen vom weiteren Bearbeiten des Tutorials abhalten. Auf bekannte Fehlerquellen kann der Benutzer im Zuge des Tutorials hingewiesen werden und die Behebung erläutert werden, aber wenn viele Fehler möglich sind, könnte das Tutorial dadurch unübersichtlich werden. Falls viele Fehlerquellen erläutert werden, würde der Benutzer viele Informationen bekommen, die er im besten Falle nicht benötigen würde und ihn stattdessen von den wichtigen Informationen ablenken. Dadurch würde der Eindruck einer normalen Anleitung entstehen.

Ein Tutorial sollte durch Hervorhebungen und Grafiken ansprechend und motivierend gestaltet sein, wobei darauf geachtet werden muss, dass es dadurch nicht unübersichtlich wird [Nixon u. a. 2009].

3.2.2. Interaktives Tutorial

Interaktive Tutorials können direkt in Programme integriert werden, sodass das Tutorial mit den Aktionen des Benutzers synchron verläuft. Über eine Textausgabe, die ebenso übersichtlich gestaltet werden sollte wie ein rein textuelles Tutorial, wird dem Benutzer der nächste Schritt erläutert, der ausgeführt werden muss. Erst wenn die entsprechende Aktion ausgeführt wurde, wird der Text für den nächsten Schritt eingeblendet, sodass der Benutzer dem Ablauf folgen muss. Zusätzlich ist es möglich, entweder alle anderen Aktionen zu blockieren, damit der Benutzer keine Fehler machen kann, oder aber die Bewegungen innerhalb des Programms zu analysieren und anhand der Position einen zusätzlichen Text einzublenden, der dem Benutzer mitteilt, wie er zurück zum Ausgangspunkt gelangen kann. Dadurch können alle Eventualitäten abgedeckt werden, ohne das Tutorial für den Benutzer unübersichtlicher zu machen.

Kapitel 4

Fragebögen

Wenn vom Aufbau von Fragebögen und den Auswirkungen bestimmter Eigenschaften auf den Prozess der Bearbeitung und die Antworten gesprochen wird, ist meistens von Meinungsumfragen die Rede, also Fragen, bei denen es kein richtig oder falsch gibt, sondern der Befragte eine ehrliche Antwort geben soll. Obwohl der Fragebogen für das Experiment nicht in diese Kategorie gehört, sind die grundlegenden Bestandteile der verschiedenen Arten von Fragebögen die selben, weshalb im Folgenden unterschiedliche Fragetypen, Skalen und Hinweise zum Aufbau beschrieben werden. Diese sind auch für das Erfragen der personenbezogenen Daten und für die Bewertung des Experiments und der Software nützlich.

Ausführlichere Informationen zu diesem Thema können [Porst 2008] entnommen werden.

4.1. Fragen

Multiple-Choice-Fragen (MC-Fragen)

MC-Fragen helfen dabei, die Gedanken des Befragten in die richtige Richtung zu lenken, da ihm vorgegeben wird, welche Antworten möglich oder erwartet sind. Dadurch wird auch die Auswertung erleichtert, weil es nicht nötig ist, Freitextantworten zu interpretieren. Gleichzeitig kann durch die Vorgabe jedoch auch eine Beeinflussung des Verhaltens eintreten, da angenommen wird, dass die möglichen Antworten sich zum Beispiel um durchschnittliches Verhalten oder realistische Möglichkeiten handeln, selbst wenn dies nicht der Fall ist und andere Intentionen hinter der Wahl der Antwortmöglichkeiten stehen.

Zudem stellen die vorgegebenen Antwortmöglichkeiten auch eine Einschränkung dar, wenn sie nicht erschöpfend gewählt wurden. Wenn der Befragte keiner Antwort zustimmt, muss er die Frage auslassen oder wird sogar dazu verleitet, eine beliebige Antwort zu wählen.

Multiple-Choice mit Einfachnennung

Bei MC-Fragen, die nur eine Antwort zulassen, müssen die Antwortmöglichkeiten disjunkt sein, da ansonsten Unklarheiten entstehen, welche Antwort zu wählen ist, wenn die gedachte Antwort in zwei Möglichkeiten beinhaltet ist.

Multiple-Choice mit Mehrfachnennung

Zu MC-Fragen mit Mehrfachnennung gehört neben den Antwortmöglichkeiten insbesondere die Angabe, wie viele Antworten maximal zugelassen sind, da nicht nur beliebige, sondern auch beschränkte Nennungen möglich sind.

Offene Fragen

Offene, oder auch freie Fragen geben dem Befragten keine Antwortmöglichkeiten vor und überlassen es ihm dadurch, die Art und Genauigkeit seiner Antwort selbst zu bestimmen. Dies kann dann problematisch sein, wenn eine Frage nicht nur zur bloßen Meinungsfeststellung gestellt wird, sondern ein Interesse an bestimmten Antworten besteht. Bei offenen Fragen kann es vorkommen, dass der Befragte in eine ganz andere Richtung denkt oder die erste Antwort aufschreibt, die ihm einfällt, ohne andere Möglichkeiten in Betracht zu ziehen.

Halboffene Fragen

Die halboffenen Fragen sind eine Mischung aus MC- und offenen Fragen, die der Problematik des nicht vollständig erschlossenen Antwortraums bei MC-Fragen entgegen wirken sollen, indem zusätzlich zu den Antwortmöglichkeiten auch eine freie Antwort gegeben werden kann. Dadurch wird die Aufmerksamkeit des Befragten auf den gewünschten Antwortraum gelenkt, ohne ihn von der Frage auszuschließen, wenn ihm tatsächlich keine der Antworten entspricht.

4.2. Skalen

Nominal-Skala

Ob die Nominal-Skala tatsächlich als Skala bezeichnet werden kann, ist umstritten, da jede Menge von Antwortmöglichkeiten, bei denen für jeden Befragten nur genau eine Antwort in Frage kommt, eine Nominal-Skala bildet. Schon einfache Ja-Nein-Möglichkeiten stellen dieser Definition nach eine Nominal-Skala dar.

Beispiel: Haben Sie eine Rot-Grün-Sehschwäche? Antwortmöglichkeiten: Ja, Nein

Ordinal-Skala

Im Gegensatz zur Nominal-Skala, bei der die Antwortmöglichkeiten nicht mehr in Verbindung stehen müssen, als dass sie sinnvolle Antworten auf die selbe Frage sind, stehen die Ausprägungen einer Ordinal-Skala in Relation zueinander. Diese Relation muss jedoch nicht gleichförmig oder genau definiert sein, ungenaue Abstufungen der Art mehr oder weniger sind ausreichend.

Beispiel: Wie hilfreich fanden Sie das Tutorial? Antwortmöglichkeiten: Sehr, ein wenig, nicht hilfreich

Intervall-Skala

Eine Intervall-Skala wiederum ist eine erweiterte Ordinal-Skala, denn auch hierbei stehen die Ausprägungen in Relation zueinander. Zusätzlich ist gefordert, dass die Abstände zwischen den Skalenpunkten gleich groß sind, was sich häufig dadurch äußert, dass nur die beiden Randpunkte benannt sind und es dem Befragten überlassen ist, sich die gleichmäßige Verteilung zu denken und entsprechend zu antworten. Dadurch kann es bei verschobenen Vorstellungen dazu kommen, dass die Befragten unterschiedliche Meinungen dazu haben, wie die Ausprägungen zu verstehen sind.

Beispiel: Wie schwer fanden Sie den Fragebogen: Antwortmöglichkeiten 1 bis 5, 1 = sehr leicht, 5 = sehr schwer

Ratio-Skala

Bei der Ratio-Skala soll nun auch das Problem der unterschiedlichen Interpretationen der gleichen Abstände behoben werden. Sie muss daher so beschaffen sein, dass es einen festen, bekannten Nullpunkt gibt, von dem aus alle anderen Skalenpunkte aus einen bekannten Abstand haben, sodass auch die Abstände zwischen den einzelnen Punkten genau definiert sind.

Beispiel: Wie alt sind Sie?

Neben der Beschriftung der Skala kann sogar die Anzahl der Skalenpunkte Einfluss auf das Verhalten der Befragten haben, da eine ungerade Anzahl andere Möglichkeiten bietet als eine gerade. Bei Fragen, deren Skala sich von einer negativen zu einer positiven Einstellung erstreckt, wird der mittlere Punkt als neutral erachtet und wird in dieser Funktion nicht nur von den Personen gewählt, deren Einstellung tatsächlich neutral ist, sondern auch von denen, die sich nicht ohne weiteres auf eine Seite festlegen können. Ohne einen solchen Mittelpunkt zwingt man die Befragten, sich zu entscheiden, nimmt aber denen, die tatsächlich neutral sind, ihre Antwortmöglichkeit und verleitet sie dazu, wahllos eine der beiden mittleren Optionen zu wählen.

4.3. Formaler Aufbau

Die Formulierung der Fragen sollte so einfach sein, wie die Thematik es erlaubt, damit es weder zu Missverständnissen noch zu Unverständnis kommt, wie es der Fall sein kann, wenn eine Frage kompliziert gestellt wird oder Fachwörter enthält, deren Kenntnis nicht bei allen Befragten vorausgesetzt werden kann. Wenn eine Frage nicht oder falsch verstanden wird, ist es wahrscheinlich, dass keine oder nicht die beabsichtigte Antwort gegeben wird.

Des Weiteren sollte verhindert werden, dass durch die Wortwahl der Frage suggeriert wird, welche Antwort gegeben werden sollte. Dieser Umstand wird zum Beispiel bei moralischen Fragen sichtbar, bei denen der Befragte durch eine provokative Formulierung eher die Antwort wählen würde, die er als moralisch richtig betrachtet, als die, die seiner Meinung entspricht, selbst wenn durch Anonymität keine Verbindung zu ihm hergestellt werden könnte. Ist die Frage hingegen wertungsfrei gestellt, sind Menschen eher bereit dazu, ihre Meinung anzugeben, selbst wenn diese nicht gesellschaftlich akzeptiert ist.

Bei Antwortmöglichkeiten, die eine Zustimmung oder Wertung erfragen, sollte darauf geachtet werden, dass die Skala von links nach rechts aufsteigend positiv ist und nicht gegenläufig, da das aufsteigende Verhalten als natürlicher wahrgenommen wird. Dies soll den Ablauf für den Befragten flüssiger gestalten, da er sich nicht darauf konzentrieren muss, wie die Antworten angeordnet sind. Aus dem gleichen Grund sollten sich MC-Fragen mit Einfach- und Mehrfachnennung visuell durch die Gestaltung der Antwortmöglichkeiten unterscheiden, damit der Befragte schon dadurch erkennen kann, was von ihm erwartet wird.

Zudem sollte darauf geachtet werden, dass die Schrift groß genug und leicht zu lesen ist und die einzelnen Fragen genug Platz haben. Dies verhindert zugleich, dass zu viele Fragen auf einmal gestellt werden, was abschreckend auf den Befragten wirken kann.

4.4. Fragebogenarten

Wie schon erwähnt, gibt es große Unterschiede zwischen den Fragebögen, die für Meinungsumfragen genutzt werden und denen, die einen Test-Charakter haben, die nicht nur die Auswirkungen des Aufbaus sondern auch die sinnvollen Möglichkeiten des Fragebogens betrifft.

Meinungsumfrage

Bei einer Meinungsumfrage geht es nicht darum, zwischen richtig und falsch zu unterscheiden, sondern herauszufinden, wie der Befragte über bestimmte Themen denkt. Das Ziel ist also, eine möglichst ehrliche Antwort zu erhalten, wozu insbesondere gehört, dem Befragten die Möglichkeit zu geben, die Antwort geben zu können, die seiner Meinung entspricht. Sind die vorgegebenen Antworten zu eingeschränkt, sodass der Befragte nicht genau das ausdrücken kann, was er möchte, kann es zu Frust und dem Abbruch der Umfrage führen oder, falls die Beendigung belohnt wird oder auf irgendeine Weise eingefordert werden kann, zu willkürlichen Antworten, was nicht im Sinne der Umfrage liegt.

Testfragebogen

Im Gegensatz zu einer Meinungsumfrage geht es bei Testfragebögen darum, herauszufinden, ob der Befragte korrekte Antworten auf die Fragen geben kann, das heißt für jede

Frage liegt eine korrekte Antwort vor, gegen die die gegebenen Antworten abgeglichen werden können. Offene Fragen können dabei gut genutzt werden, um zu verhindern, dass der Getestete durch Raten richtige Antworten gibt und um herauszufinden, ob er auch auf die richtige Antwort kommt, wenn diese ihm nicht durch die Antwortmenge einer MC-Fragen vor Augen gehalten wird. Es bleibt jedoch der Nachteil der erschwerten Auswertung, die bei MC-Fragen einfacher ist, da sofort ersichtlich ist, ob die richtige Antwort gewählt wurde.

Werden MC-Fragen genutzt, sind nur Nominal-Skalen und Ratio-Skalen sinnvoll, da Ordinal- und Intervall-Skalen zu viel Spielraum lassen, um garantieren zu können, dass ein Teilnehmer, der die richtige Antwort weiß, die Skala korrekt interpretiert und dadurch in der Lage ist, die entsprechende Antwort auszuwählen.

Kapitel 5

Ansatz

In diesem Kapitel werden detailliertere Anforderungen betrachtet und der Ansatz beschrieben. Des Weiteren werden Entscheidungen für die spätere Umsetzung getroffen. Dabei wird zuerst der Tutorialmodus und anschließend der Fragebogenmodus betrachtet.

5.1. Tutorialmodus

Zuerst muss festgestellt werden, welche Funktionen ExplorViz bietet und wie diese Funktionen aneinander gereiht werden können, um einen fließenden Ablauf zu erhalten, und dadurch das Verständnis des Benutzers zu fördern. Außerdem muss Entschieden werden, wie die Texte des Tutorials verwaltet werden sollen.

Eine wichtige erste Entscheidung ist, ob ein Tutorial, das einen vorgegebenen Ablauf hat und unter idealen Bedingungen nicht verändert werden muss, fest implementiert oder ein modularer Ansatz gewählt werden sollte. Ein fester Ablauf hat den Vorteil, dass nur eine Fallunterscheidung notwendig sind, da das Verhalten vollständig durch die Nummer des aktuellen Tutorialschritts vorgegeben ist und nicht durch den Inhalt des Schrittes bestimmt werden muss. Der Nachteil eines solchen Herangehens ist aber, dass Änderungen am Programm selbst große Änderungen am Tutorial nötig machen können und das Hinzufügen von weiteren Schritten bei Weiterentwicklung des Programms neue Arbeit erfordert. Da ExplorViz sich zu Beginn dieser Arbeit noch in Entwicklung befand und nicht absehbar war, wann diese beendet sein wird, wurde das modulare Vorgehen gewählt, in dem das Verhalten eines Tutorialschritts nicht durch die Nummer sondern die zu dem Schritt gespeicherten Informationen festgelegt wird.

5.1.1. Zu erklärende Funktionen

Wie in Abschnitt 2.1 erläutert, bietet ExplorViz zwei verschiedene Ansichten an, um die verschiedene Ebenen einer Softwarelandschaft darstellen. Um eine Anwendung genauer zu betrachten, eine Knotengruppe zu öffnen oder den Inhalt einer Komponente anzeigen zu lassen, muss das jeweilige Element doppelt angeklickt werden. Auf die selbe Weise werden Elemente auch wieder geschlossen.

Wird stattdessen mit der rechten Maustaste auf eine Komponente geklickt, wird ein Kontextmenü geöffnet, das Interaktionen mit der Komponente ermöglicht, wie das Anzei-

gen des Sourcecodes im Codeviewer in der Anwendungsebene. Außerdem sollen von dort aus Änderungen am Monitoring vorgenommen und Applikationen gestartet und gestoppt werden können. Da sowohl für das Experiment als auch das Tutorial keine Live-Daten genutzt werden, wird diese Funktion nur im Zusammenhang mit dem Kontextmenü erwähnt, ohne weiter darauf einzugehen.

Von der Landschaftsansicht kann man in die Anwendungsansicht wechseln, indem die gewünschte Anwendung zweimal angeklickt wird. Um von dort aus wieder zurück zur Landschaftsansicht zu kommen, wird in der Anwendungsansicht ein Button eingeblendet.

Die Ansicht kann mit der Maus verschoben werden und die Zoomstufe mit dem Mausrad angepasst werden.

5.1.2. Vorgegebene Softwarelandschaft als Grundlage

Um die Implementierung eines Tutorials zu erleichtern und sicherzustellen, dass für jeden Schritt des Tutorials eine entsprechende Aktion möglich ist, wird eine vorgegebene Softwarelandschaft als Anschauungsobjekt genutzt. Es wäre zwar möglich, durch Analyse der Landschaft festzustellen, welches System sich am besten eignet, um das Ein- und Ausklappen von Knotengruppen und die Anwendungsebene zu erläutern, aber wenn es keine ausklappbare Knotengruppe gäbe, müsste der Schritt übersprungen werden, sodass das Tutorial nicht mehr bei jeder Benutzung identisch wäre.

Als Beispiellandschaft wurde ein Teil des PubFlow-System [*PubFlow*] mit den dazu gehörigen Komponenten gewählt, da es sich aus mehreren Systemen mit Datenbanken und Interfaces zusammensetzt. Durch die mittlere Größe der Landschaft kann der Benutzer einen realistischeren Eindruck davon gewinnen, wie die Benutzung von ExplorViz funktioniert, als wenn es nur eine kleine Landschaft wäre. Gäbe es nur zwei Systeme mit ein bis zwei Anwendungen, würde nicht deutlich werden, weshalb es von Nutzen ist, Systeme und Anwendungsknoten ein- und ausklappen zu können.

Die Veränderung der Landschaft über die Zeit hinweg wird durch erhöhte Kommunikation zwischen PubFlow und OceanRep dargestellt, indem dieselbe Softwarelandschaft mit anderen Kommunikationswerten eingelesen wird, sobald der entsprechende Tutorialschritt erreicht wird.

5.1.3. Festlegen des Ablaufs

Aufgrund der Größe der Beispiellandschaft ist es möglich, verschiedene Funktionen und ihren Nutzen zu erklären, ohne sich auf ein System und deren Applikationen zu beschränken. Durch eine zu starke Fokussierung auf einen Teil des Systems würde der Benutzer den Nutzen der Möglichkeit, unbenötigte Komponenten einzuklappen nicht erkennen. Zusätzlich soll die Notwendigkeit, in der Landschaft zu scrollen, dazu beitragen, dass der Benutzer nicht ohne zu lesen die gerade sichtbaren Komponenten anklickt, bis etwas passiert, was ein mögliches Verhalten bei einer geringen Anzahl Möglichkeiten wäre. Der Ablauf des Tutorials sieht wie folgt aus:

```
Erklärung von Systemen und Kommunikation
Schließen eines Systems
Öffnen eines Systems
Erklärung von Knoten, Knotengruppen und Anwendungen
Öffnen einer Knotengruppe, Hinweis auf das +
Schließen einer Knotengruppe Hinweis auf das -
Öffnen einer Applikation
Erklärung der Anwendungsansicht
Öffnen eines Pakets
Hovern über Paket
Öffnen eines weiteren Pakets
Linksklick auf eine Klasse
Erklärung der ein- und ausgehenden Kommunikation
Hovern über Kommunikation
Öffnen eines Pakets
Klick auf Klasse x 3 zum Folgen der Kommunikation
Erklärung von explorativem Raten
Linksklick auf Kommunikation
Auswählen eines Traces
Nutzen des Trace Replayers (Start, Stop, Next)
Verlassen des Trace Replayers
Kontextmenü einer Klasse öffnen
Codeviewer öffnen
Codeviewer verlassen
Schließen einer Komponente
Rückkehr zur Landschaftsansicht
Timeshift mit Erklärung und Hinweis auf den Pause-Knopf
```

Neben den interaktiven Schritten soll es möglich sein, ausschließlich beschreibende Schritte anzulegen. Grundsätzlich sind solche Schrittte nicht notwendig, da der Text in einen interaktiven Schritt integriert werden könnte, aber dadurch können die Texte der einzelnen Schritte sehr lang werden. Wie in Abschnitt 3.1 festgestellt, wird Computerbenutzern eine Abneigung gegenüber längeren Texten nachgesagt. Die Aufteilung in kleinere Schritte soll dabei helfen, dem Benutzer selbst dann so viele Informationen mitzuteilen wie möglich, wenn er jeden Text nur überfliegt, anstatt ihn aufmerksam zu lesen.

Weiterhin soll von Anfang an Wert darauf gelegt werden, dass neue Funktionen leicht in das Tutorial integriert werden können. Es soll daher kein festgelegter Ablauf sein, sondern nach Bedarf veränderbar und erweiterbar sein. Dies muss bei der Implementierung berücksichtigt werden.

5.1.4. Speichern der Texte

Bezüglich der Frage, auf welche Weise die Texte des Tutorials hinterlegt werden sollen, gibt es drei Umstände zu beachten:

- ▻ Wie leicht ist es, Texte für eine zusätzliche Sprache hinzuzufügen?
- ▻ Welche Möglichkeiten zur Formatierung der Texte sollen möglich sein?
- ▻ Wie rechenintensiv ist das Einlesen der Texte? Muss zwischen Formaten konvertiert werden?

Eine Möglichkeit, um das Einlesen von Texten aus einer externen Quelle zu umgehen, besteht darin, die Texte direkt in das Programm zu schreiben und sie als eine Liste von Strings zu speichern. Für jede Sprache würde eine solche Liste angelegt und in eine Liste von Listen eingefügt werden, über die die Sprachauswahl erfolgen würde. Der Nachteil in einem derartigen Vorgehen liegt darin, dass es aufwändig und umständlich ist, die Texte des Tutorials zu verwalten.

Eine einfache Verwaltung kann mithilfe einer Datenbank erreicht werden, in der alle Texte, zusammen mit den Informationen über die Sprache und ihrer Position innerhalb des Tutorials gespeichert werden. Eine Datenbank aufzubauen, die nur für ein Tutorial genutzt wird, erscheint allerdings nur begrenzt sinnvoll. Da eine Datenbank für die Benutzerverwaltung besteht, könnte dort allerdings eine zusätzliche Tabelle für die Textverwaltung eingefügt werden.

Zusätzlich bietet sich an, die Texte aus einer Datei auszulesen, die auf dem Server gespeichert wird. Eine Möglichkeit dabei wäre, für jede Sprache eine Datei anzulegen und die Texte der Reihe nach und von Trennzeichen unterbrochen zu speichern. Beim Verstellen der Sprache würde dann nur der Name der zu lesenden Datei geändert werden. Dabei ist zu bedenken, wie das Trennzeichen gewählt wird, um die Formatierung der Texte und die benutzbaren Zeichen so wenig wie möglich einzuschränken. Um die Problematik von Trennzeichen und das Zuordnen der einzelnen Texte zu den Schritten des Tutorials zu vereinfachen, können die Texte stattdessen in einzelnen, durchnummerierten Dateien abgelegt werden. Die Dateien der einzelnen Sprachen werden zur besseren Übersicht auf entsprechend benannte Ordner verteilt. Diese Variante wird aufgrund der genannten Vorteile umgesetzt.

Weiterhin ist zu entscheiden, ob alle Texte des Tutorials zu Beginn eingelesen werden sollen, also einmal eine größere Anfrage an den Server gesendet werden soll, oder der Text für jeden Tutorialschritt abgerufen werden soll, wenn er benötigt wird. Beide Möglichkeiten sind aufgrund der Verwaltung ohne zusätzlichen Aufwand nutzbar. Aufgrund der Spracheinstellungsmöglichkeit auf dem Server, die auch im Laufe eines Tutorials geändert werden könnte, bietet es sich an, die Texte nacheinander zu laden, sodass die geänderte Sprachwahl beim nächsten Schritt automatisch berücksichtigt wird, und nicht das gesamte Tutorial neu geladen werden muss. Dies würde eine Benachrichtigung bei der

Veränderung der Sprache benötigen, während die Änderung beim Laden der einzelnen Texte automatisch im nächsten Schritt berücksichtigt wird.

5.2. Fragebogenkonzept

Wie auch schon beim Tutorialmodus bietet es sich beim Fragebogenkonzept an, bestimmte Entscheidungen zu treffen, bevor mit der Umsetzung begonnen wird. Dadurch soll verhindert werden, dass die Implementierung nachträglich an größere, konzeptuelle Entscheidungen angepasst werden muss, die nicht bedacht worden sind. Stattdessen sollen diese Entscheidungen von vornherein berücksichtigt werden. Dazu gehört die Auswahl der unterstützten Fragetypen und die Frage nach dem Format, in dem die Fragen und Antworten gespeichert werden sollen. Außerdem muss die Problematik von Fragen beachtet werden, die nur zu bestimmten Zeitpunkten richtig beantwortet werden können und daher die Benutzung der Timeshift-Funktion erfordern. Im Folgenden werden die Vor- und Nachteile der verschiedenen Fragetypen diskutiert.

5.2.1. Mögliche Fragetypen

Damit ein geeignetes Format zum Speichern der Fragen ausgewählt werden kann, muss zuvor festgelegt werden, welche Arten von Fragen benötigt werden. Ansonsten kann es passieren, dass ein Format ausgewählt wird, dass die Möglichkeit mehrerer richtiger Antworten nicht unterstützt, was die Nutzung einschränken würde, falls solche Fragen gewünscht sind.

Freitextfragen Problematisch ist bei Freitextfragen die Auswertung der Antworten, da durch Tippfehler oder Ungenauigkeiten eine eigentlich richtige Antwort bei einem automatisierten Vergleich mit der richtigen Antwort nicht als gleich erkannt werden würde. Die Auswertung muss daher entweder von einem Korrektor bewertet werden, oder es muss ein Vergleichsalgorithmus genutzt werden, der nicht nur auf identische Zeichenketten sondern auch auf Ähnlichkeit prüft. Schon die Akzeptanz von einem Unterschied in nur einem Zeichen könnte aber wiederum falsche Antworten als richtig anerkennen, falls in dem Softwaresystem Komponenten auftauchen, die sich beispielsweise nur durch einen Buchstaben unterscheiden.

Multiple Choice-Fragen MC-Fragen haben den Vorteil, dass es nicht zu Schreibfehlern kommen kann. Des Weiteren geben sie dem Benutzer einen Anhaltspunkt zur Beantwortung, da er die Frage nicht nur durch das Herausfinden der richtigen Antwort, sondern auch durch das Ausschließen der falschen Antworten lösen kann. Das kann allerdings dazu führen, dass Benutzer, die sich nicht ganz sicher sind, versuchen die Antwort zu erraten.

Bei den personenbezogenen Fragen, die zu Beginn des Experiments gestellt werden sollen und den Meinungsfragen, die anschließend folgen, handelt es sich auch um Multiple

Choice Fragen. In deren Fall gibt es jedoch keine universell richtige Antwort, da die Wahl vollkommen vom Nutzer abhängt. Des Weiteren unterscheidet sich die Größe des Antwortraums bei diesen Fragen sehr, da die Frage nach dem Alter viele Antworten zulässt, während die Frage nach dem Vorhandensein einer Rot-Grün-Sehschwäche nur zwei mögliche Antworten hat. Zusätzlich ist es bei bestimmten Fragen hilfreich, wenn die Antwortmöglichkeiten näher erläutert werden können, damit alle Benutzer die gegebene Skala gleich interpretieren. Ein weiterer Unterschied liegt in dem Nutzen, der durch das gleichzeitige Anzeigen der Antwortmöglichkeiten gewonnen wird. Bei der Frage nach einer richtigen Antwort ist es hilfreich, alle Antworten sehen zu können, um sie sofort mit der Visualisierung vergleichen zu können, während eine Meinungsfrage sofort beantwortet werden kann und daher Darstellungen wie eine Combobox ausreichend sind, die nur auf einen Klick hin alle Antworten anzeigen. Aus diesen Gründen bietet es sich an, diese statistischen Fragen anders zu behandeln als die MC-Fragen des eigentlichen Fragebogens. Dies wird bei der Auswahl des Speicherformats (siehe Unterabschnitt 5.2.2) berücksichtigt.

Multiple Choice-Fragen mit mehreren richtigen Antworten Eine Unterkategorie der MC-Fragen sind solche, bei denen es nicht zwangsläufig nur eine richtige Antwort gibt, sondern es dem Benutzer überlassen bleibt, herauszufinden, ob nur eine oder sogar alle Vorgaben korrekt sind. Das Erraten der richtigen Lösung wird dadurch erschwert. Im Gegenzug muss bei der Auswertung der Antworten auch darauf geachtet werden, dass teilweise richtige Antworten, also solche, bei denen eine richtige Antwort ausgelassen oder zusätzlich eine falsche Antwort gegeben wurde, nicht als richtig anerkannt werden.

Halboffene Fragen werden nicht benötigt, da aufgrund der Tatsache, dass es eine richtige Antwort gibt, bei MC-Fragen davon ausgegangen werden kann, dass die richtige Antwort in der Auswahl vorhanden ist.

5.2.2. Speicherformat der Fragen

Eine der Herausforderungen beim Fragebogenkonzept ist das Finden eines geeigneten Speicherformats für die Fragen. Da eine GUI zum Erstellen und Bearbeiten der Fragen nur eine optionale Anforderung ist, muss ein Format gewählt werden, das neben den Anforderungen für die Nutzbarkeit insbesondere auch für Menschen leicht lesbar sein. Dabei sind folgende Fragen von Interesse:

- Wie umständlich ist das Bearbeiten des Fragekatalogs?
- Wie gut werden die verschiedenen Fragetypen unterstützt?
- Wie leicht lassen die Fragen sich einlesen?

Ebenso wie für die Texte des Tutorials ist eine Datenbank ein geeignetes Medium zum Speichern der Fragen. Diese lassen sich leicht abfragen und sind auch von Menschen lesbar, sodass Änderungen ohne viele Probleme vorgenommen werden können. Einzelne

Anpassungen von Fragen sind auch nicht sehr umständlich, für einen ganzen Fragenkatalog muss jedoch entweder jedes Feld einzeln bearbeitet oder eine insert-Funktion geschrieben werden, die alle Felder korrekt füllt. Dabei können leicht Fehler gemacht werden, die zu finden in einer langen Fragenliste umständlich ist. Dazu kommt, dass eine Beschränkung der möglichen und richtigen Antworten für MC-Fragen gegeben ist, beziehungsweise bei jedem Überschreiten der bisherigen Anzahl an Spalten die Tabelle angepasst werden muss.

Da das CSV-Format eine andere Art der Darstellung einer Tabelle ist, entstehen ähnliche Probleme wie bei der Nutzung einer Datenbank. Neben der für Menschen nicht gut lesbaren und daher umständlich zu bearbeitenden Form ergibt sich außerdem das Problem des Trennzeichens, da Fragestellungen alle Satzzeichen enthalten können. Es müssten daher zusätzliche Vorkehrungen getroffen werden, damit die Trennzeichen beim Einlesen korrekt erkannt werden.

Eine andere Möglichkeit besteht darin, ein eigenes Format zu benutzen, dass für Menschen leicht lesbar ist. Dieses Format muss beim Einlesen korrekt geparst werden und benötigt dadurch mehr Rechenleistung als das Auslesen aus einer Datenbank oder Trennung anhand eines Zeichens. Außerdem muss beim Erstellen der Fragen darauf geachtet werden, das vorgegebene Format einzuhalten, da das Parsen andernfalls fehlschlägt. Ein solches Format hat jedoch den Vorteil, dass es darauf ausgelegt werden kann, eine beliebige Anzahl von Antworten zu unterstützen, ohne Anpassungen vornehmen zu müssen.

Als Speicherformat wird aufgrund der leichten Bearbeitbarkeit ein eigenes Format (siehe Listing 5.1) genutzt. Jede Frage besteht dabei aus genau fünf Zeilen, Zeilenumbrüche innerhalb des Fragetextes sind daher nicht erlaubt. Sollte es hingegen für Anwendungszwecke sinnvoll werden, mehrere Fragebögen zu verwalten, könnte in Zusammenhang mit einer entsprechenden GUI eine Datenbank sinnvoller sein, da diese bessere Editierungsmöglichkeiten bietet als eine reine Textdatei.

Listing 5.1. Format der Fragen

```
Question: <Fragentext>
Answers: <Antwort 1>, <Antwort 2>, ...
Correct Answers: <richtige Antwort 1>, ...
Free Answers: <Anzahl Eingabefelder bei freien Fragen>
Processing time: <Zeit in Minuten, bis der Timer rot wird>
Timestamp: <Zeitstempel bis zu dem die Frage aktuell ist>
```

Die einführenden und abschließenden Fragen (statistische Fragen) werden in einer zweiten Datei abgelegt, die in sechs Abschnitte unterteilt ist, die angeben, welche Fragen zusammen angezeigt werden. Für die Fragen stehen verschiedene Formate zur Verfügung, abhängig davon, welcher Inputtyp für die Frage genutzt werden soll, siehe Listing 5.2. Dies beruht darauf, da je nach Inputtyp unterschiedliche Attribute zur Verfügung stehen, die zur Konfiguration genutzt werden können. Dies ist für den Hauptteil des Fragebogens nicht notwendig, da dieser mit vier verschiedenen Inputformaten auskommt, die eindeutig durch die Anzahl der möglichen und richtigen Antworten bestimmt werden können.

Listing 5.2. mögliche Formate für statistische Fragen

```
Choice: <Fragetext Combobox>
Answers: <Antworten durch Komma getrennt>
Tooltip: <Tooltiptext, wird neben Frage angezeigt>
Binary: <Fragetext Radiobuttons>
Yes: <Beschriftung für Antwort 1>
No: <Beschriftung für Antwort 2>
```

Eine andere Möglichkeit wäre gewesen, zu jeder Frage alle Attribute aufzulisten, nicht zutreffende leer zu lassen und anhand der gegebenen Attribute zu bestimmen, welche Art von Eingabe angezeigt werden soll, wie es auch bei den Fragen des Hauptteils der Fall ist. Durch die vielen Möglichkeiten würde dies aber sowohl das Parsen als auch das Editieren der Fragen komplizierter gestalten.

5.2.3. Speicherformat der Antworten

Die Antworten, die von den Benutzern gegeben werden, werden auf dem Server gespeichert, um von dort exportiert und ausgewertet zu werden. Da nicht vorgesehen ist, dass die Antworten manuell bearbeitet werden, ist die Lesbarkeit für Menschen bei der Auswahl des Formats vernachlässigbar. Beachtet werden stattdessen folgende Kriterien:

- Wie aufwändig ist die Konvertierung in das Speicherformat und wie gut eignet sich das Format für die nötigen Informationen?
- Wie leicht lassen sich die Daten exportieren um mit R Skript ausgewertet zu werden?

Eine Möglichkeit wäre auch hier, eine Datenbank zu nutzen, in der die Antworten verwaltet werden. Diese hätte den Vorteil, dass auch MC-Fragen mit mehreren richtigen Antworten ohne gesonderte Behandlung gestellt und deren Antworten gespeichert werden können. Für die spätere Auswertung würden die Daten von dort exportiert und in ein geeignetes Format wie das CSV-Format konvertiert werden müssen.

Da eine Konvertierung zum CSV-Format letztendlich auf jeden Fall durchgeführt werden muss, kann auf den Zwischenschritt verzichtet werden. Stattdessen können die Antworten sofort im Server in einer CSV-Datei abgelegt werden (Format siehe Listing 5.3). Hierbei ist allerdings zu überlegen, ob es sinnvoll ist, die Datei für jede Antwort zu editieren oder ob es sinnvoller ist, die Antworten erst clientseitig zu sammeln und nach der letzten Frage oder dem Ablaufen der Zeit an den Server zu senden, sodass alle Antworten zusammen weggeschrieben werden können. Anschließend muss die Datei zur Auswertung nur noch exportiert, nicht aber konvertiert werden.

5.2.4. Fragen mit bestimmten Zeitpunkt verbinden

Da mit ExplorViz nicht nur Momentaufnahmen einer Softwarelandschaft visualisiert werden können, sondern insbesondere auch die Veränderungen über die Zeit hinweg

Listing 5.3. Formatbeispiel für Antworten

```
UserID, FrageID, gebrauchte Zeit, Bearbeitungsstart, Bearbeitungsende,
Antwort1, Antwort2, ...
```

dargestellt werden, kann es vorkommen, dass Fragen nur zu bestimmten Zeitpunkten sinnvoll zu beantworten sind. Würden die Fragen nun der Reihe nach, aber nicht auf die Vorkommnisse in der Softwarelandschaft abgestimmt gestellt werden, kann es passieren, dass ein Benutzer eine Frage erhält, die er noch nicht oder nicht mehr beantworten kann, je nachdem, wie schnell er die vorherigen Fragen beantwortet hat.

Um diesem Problem entgegenwirken zu können, muss die vorliegende Softwarelandschaft bekannt sein, das heißt es kann sich nicht um eine live generierte Landschaft handeln sondern muss ein vorher aufgezeichnetes Verhalten sein. Dadurch ist es möglich, festzulegen, welche Fragen während welcher Zeiten beantwortet werden müssen und durch die Timeshift-Funktion, die das Anhalten der Visualisierung ermöglicht, diese Verknüpfung zu erzwingen. Dazu wird zu einer Frage gespeichert, bis zu welchem Zeitpunkt sie gültig ist, und die Visualisierung nur bis zu dem Punkt abgespielt.

Kapitel 6

Implementierung

In diesem Kapitel wird beschrieben, wie der Tutorialmodus und der Fragebogenmodus aufbauend auf dem Ansatz und den Konzepten und Entscheidungen aus Kapitel 5 implementiert wurden.

6.1. Tutorialmodus

Nachdem die zu treffenden Entscheidungen erläutert und getroffen wurden, wird im Folgenden beschrieben, wie der Tutorialmodus in ExplorViz integriert wurde, und wie die einzelnen Schritte des Tutorials ablaufen.

Im Rahmen der Implementierung wird der Begriff *Komponente* für alle Bestandteile der Softwarelandschaft benutzt und ist nicht zu verwechseln mit den Komponenten einer Applikation.

6.1.1. Integration in ExplorViz

Der ursprüngliche Plan sah vor, Vererbung zu benutzen, um die tutorialspezischen Funktionen in die Darstellung und das Verhalten der Visualisierung zu integrieren. Dadurch sollten die verschiedenen Modi voneinander getrennt werden, ohne Code zu verdoppeln, und die Modularität gewährleistet werden. Die neuen Klassen hätten nur die Methoden überschrieben, die sich unterschiedlich verhalten, während gleiches Verhalten weiterhin von den ursprünglichen Methoden gesteuert wird. Dies wurde jedoch dadurch verhindert, dass viele Klassen private, statische Methoden enthalten, welche nicht vererbt werden können. Es ist zwar möglich, statische Methoden einer Superklasse durch eine Methode mit gleicher Signatur zu verstecken, aber wenn aus dieser Klasse heraus statische, private Methoden der Superklasse aufgerufen werden sollen, stellt die nicht-Vererbung ein Problem da.

Stattdessen wird an den Stellen, an denen Vererbung nicht genutzt werden kann, eine boolesche Variable genutzt, die als Indikator für die Befindlichkeit im Tutorialmodus agiert. Dadurch ist es möglich, in den Methoden eine Fallunterscheidung zu machen und entsprechend des Modus unterschiedliches Verhalten zu implementieren.

Zu den vorzunehmenden Änderungen gehören das Laden der Tutoriallandschaft anstelle der Livedaten des normalen Betriebs, das Deaktivieren von Funktionen und das Erstellen

eines Pfeils, der die textuelle Anweisung unterstützt. Weiterhin muss beim Interagieren mit der Landschaft überprüft werden, ob der Benutzer die korrekte Aktion ausgeführt wird.

Außerdem ist das Konfigurationsmenü um eine Option erweitert worden, die es ermöglicht, die Sprache des Tutorials zu verstellen. Diese Einstellung wird auf dem Server gespeichert und gilt daher für alle Nutzer.

6.1.2. Starten des Tutorials

Das Tutorial soll automatisch gestartet werden, wenn sich ein Benutzer zum ersten Mal bei ExplorViz einloggt. Zu diesem Zweck enthält die Datenbank eine Spalte `firstLogin`, die nach dem Einloggen abgefragt werden kann. Handelt es sich um den ersten Login des Benutzers, wird er zum Tutorial weitergeleitet, nach dessen Beendigung der Wert auf `false` gesetzt und beim Logout in die Datenbank geschrieben wird.

Die Seite, die nach dem Einloggen aufgerufen wird, wird durch die `callFirstPage`-Methode bestimmt. Normalerweise wird von dort die normale ExplorViz-Seite aufgerufen. Wird diese Methode jedoch aufgerufen, wenn ein Benutzer eingeloggt ist, der zuvor noch nicht eingeloggt war, wird stattdessen die Tutorialseite geladen. Da es länger dauern kann, die Benutzerdaten vom Server abzufragen, als die Seite aufzubauen, wird die `callFirstPage`-Methode in der `onSuccess`-Methode des Benutzers aufgerufen, um eine Race Condition zu verhindern.

Zusätzlich kann das Tutorial aus dem Menü heraus gestartet werden. Dazu wird ein weiterer Punkt im Menü eingefügt, der entsprechend der anderen Menüpunkte in der Hauptklasse mit der `TutorialPage` verknüpft wird.

Da das Tutorial mit einer vorgegebenen Landschaft arbeitet, muss der `Landscape-Ex-changeManager` angepasst werden, was durch eine Fallunterscheidung sowie eine abgeleitete `ExchangeService`-Klasse geschieht, die beim Aufruf eine statische Landschaft übermittelt. Dieser neue Exchange-Service sorgt auch dafür, dass ab einem durch den Tutorialablauf festgelegten Zeitpunkt eine leicht veränderte Version der Landschaft gesendet wird, sodass die Auswirkungen des Timeshift sichtbar sind.

6.1.3. Tutorialschritt

Ein wichtiger Punkt, um den Ablauf des Tutorials zu gewährleisten, ist zu speichern, in welchem Schritt der Nutzer sich gerade befindet, da das Verhalten des Tutorials von diesem Wert abhängt. Dadurch wird der Text der Dialogbox bestimmt sowie festgelegt, welche Aktionen den Übergang zum nächsten Schritt auslösen.

Step-Klasse

Der Ablauf des Tutorials wird durch eine Liste von Schritten vorgegeben, die in der auszuführenden Reihenfolge in der Configuration-Klasse des Servers gespeichert werden. Ein solcher Schritt enthält ein Flag, das markiert, ob der Schritt sich auf eine Komponente

oder eine Verbindung bezieht. Verbindungen werden durch den Namen der ausgehenden und der eingehenden Komponente identifiziert, Komponenten werden nur über ihren Namen identifiziert. Über boolesche Variablen wird angegeben, ob die geforderte Aktion durch einen Doppel-, Rechts- oder Linksklick oder Hovering ausgelöst wird. Dabei muss darauf geachtet werden, dass nur Aktionen auf Komponenten gefordert werden, für die auch ein entsprechender MouseClickHandler existiert. Weitere boolesche Variablen werden genutzt, um den Übergang von der Application- zur Landschaftsansicht, die Nutzung der Timeshift-Funktion und Aktionen innerhalb von Dialogen zu fordern.

Zusätzlich sind rein informative Texte möglich, zu denen keine Handlung ausgeführt werden muss. Stattdessen wird ein Button in der Dialogbox eingeblendet, der den Übergang zum nächsten Schritt auslöst. Da auch informative Texte durch graphische Hinweise begleitet werden können, ist es möglich, den Namen einer Komponente anzugeben, auf die ein Pfeil deuten soll.

Durch diese Vielzahl unterschiedlicher Tutorialschritte wird es nötig, sehr viele Flags beim Erstellen eines Schritts zu setzen. Die Möglichkeiten dafür waren entweder ein einzelner Konstruktor mit einer entsprechend großen Anzahl Parameter, oder verschiedene Konstruktoren. Da es bei Erweiterungen von ExplorViz leicht möglich sein soll, das Tutorial anzupassen, wäre ein einzelner Konstruktor unhandlich. Stattdessen wurden die Schritte in logische Gruppen unterteilt. Für jede dieser Gruppen existiert ein Konstruktor, sodass nur so viele Parameter benötigt werden, wie nötig sind, um die Schritte der Gruppe zu identifizieren.

Auf Interaktionen reagieren

Für jede Komponente der Landschaft und Applikationen existieren Handler, um dem Benutzer unterschiedliche Interaktionen zu ermöglichen. In diesen Handlern wird ein Methodenaufruf ergänzt, der den Namen der Komponente und die Art des Handlers beinhaltet. Dasselbe Prinzip wird in Methoden genutzt, die durch Interaktionen mit einem Dialog oder Kontextmenü aufgerufen werden. Diese Methode testet, ob sich der Benutzer gerade im Tutorialmodus befindet. Ist dies der Fall, wird geprüft, ob es sich bei der ausgeführten Interaktion tatsächlich um die Aktion handelt, die den nächsten Tutorialschritt auslöst. Diese Überprüfung ist nötig, da es für bestimmte Anweisungsabfolgen nötig sein kann, eine Aktion auch in einem anderen Schritt zu erlauben (siehe Abschnitt 6.1.3) Ist dies der Fall, wird ein Tutorialschritt ausgeführt. Das bedeutet, dass der Schrittzähler inkrementiert, der neue Text geladen und für die Überprüfung der Interaktionen der nächste Schritt betrachtet wird - dies geschieht in der incStep-Methode.

Die incStep-Methode sondern ruft die getText-Methode auf, die per RPC-GWT den TutorialService kontaktiert. Ist der Callback erfolgreich, wird eine JavaScript-Funktion ausgeführt, die den Inhalt des Tutorialdialogs (siehe Unterabschnitt 6.1.5) mit dem neuen Text überschreibt. Die Dialoggröße wurde so festgesetzt, dass ausreichend Platz für kurze Texte vorhanden ist. Sollen deutlich längere Texte genutzt werden, müsste die Einstellung angepasst werden, wobei zu empfehlen ist, die Höhe, nicht jedoch die Breite zu verändern.

Da widescreen Monitore weit verbreitet sind, ist ein hoher, schmaler Dialog weniger störend als ein kurzer, breiter.

Verhindern falscher Interaktionen

Um den Ablauf zu erleichtern und zu verhindern, dass durch unerwartete Aktionen eine Sackgasse erreicht wird, d.h. der Benutzer nicht weiß, wie er eine Ansicht wieder verlässt, werden im Verlauf des Tutorials nur die MouseClickHandler aktiviert, die zur Vervollständigung des aktuellen Tutorialschritts notwendig sind.

Da beim Aufbau der Landschaft normalerweise zuerst alle bisher gesetzten Handler entfernt werden, um Interaktionen mit nicht mehr angezeigten Komponenten zu verhindern, und anschließend für alle angezeigten Komponenten neue Handler gesetzt werden, gibt es zwei Möglichkeiten, wie dafür gesorgt werden kann, dass nur ein bestimmter Handler gesetzt wird: Nach dem Hinzufügen der Handler wird eine Methode aufgerufen, die alle Handler bis auf den gesuchten wieder entfernt werden, oder das Hinzufügen wird so verändert, dass im Tutorialmodus nur der gesuchte Handler gesetzt wird.

Zuerst alle Handler hinzuzufügen, um sie hinterher wieder zu entfernen, würde unnötige Arbeit verursachen. Stattdessen werden die createInteraction Methoden um eine Fallunterscheidung ergänzt, die überprüft, ob die gerade betrachtete Komponente für das Tutorial einen Handler benötigt. Ist dies der Fall, kann darauf verzichtet werden, die Handler der Kind-Komponenten zu setzten. Bei Komponenten mit mehreren Handlern werden die Flags für Rechts- Links- und Doppelklick und Hovern überprüft, um nur den korrekten und nicht alle Handler der Komponente zu setzen.

Bei Interaktionen, deren Ausfühung sich nicht so leicht verhindern lässt, wird stattdessen die aufgerufene Methode um eine If-Abfrage erweitert, sodass der darin befindliche Code nicht ausgeführt wird, wenn das Tutorial aktiv aber der Benutzer im falschen Schritt ist.

Bestimmte Funktionen von ExplorViz können allerdings nur in einer festen Reihenfolge ausgeführt werden, wie zum Beispiel der Aufruf des Codeviewers aus dem Kontextmenü heraus. Würde ein Benutzer nun das Kontextmenü öffnen und damit in den nächsten Schritt übergehen, das Kontextmenü dann aber schließen, ohne den Codeviewer zu öffnen, könnte er das Tutorial nicht weiter führen. Es muss daher dafür gesorgt werden, dass bestimmte, notwendige Handlungen auch ausgeführt werden können, wenn sie nicht direkt für den aktuellen Schritt benötigt werden. Zu diesem Zweck enthält die Klasse `Experiment` ein `lastSafeStep`-Attribut. In diesem Attribut wird der letzte Schritt gespeichert, den der Benutzer sicher ausgeführt haben muss. Dadurch ist es möglich, an kritischen Stellen nicht nur den aktuellen Schritt abzufragen um die Handlung zu erlauben, sondern auch zu prüfen, ob sie für den safeStep notwendig ist. Dadurch wird dafür gesorgt, dass der Benutzer immer in der Lage ist, den nächsten Tutorialschritt zu erreichen.

6.1.4. Laden des Textes

Die Texte des Tutorials werden erst dann geladen, wenn sie gebraucht werden, sodass eventuelle Änderungen der Sprache im Laufe eines Tutorials automatisch beim nächsten Schritt angewendet werden. Stattdessen wäre es auch möglich, alle Texte zu Beginn zu laden und das Observer-Pattern zu benutzen, um über eine Veränderung der Sprache informiert zu werden und die Texte erneut zu laden, aber da es aufgrund der gewählten Datenhaltung leicht ist, nur einzelne Texte zu laden, wird diese Methode bevorzugt.

Um den aktuell benötigten Text zu laden, müssen sowohl die Sprache als auch die Nummer des Tutorialschritts bekannt sein, also intern gespeichert werden, da sich der Dateipfad der zu lesenden Datei aus diesen beiden Komponenten zusammensetzt und das Format *sprache/schritt.txt* hat. Dabei ist darauf zu achten, dass die Dateinamen keine führenden Nullen enthalten, da der interne Schrittzähler eine int Variable ist und unverändert in den Pfad aufgenommen wird. Alle Zeilen dieser Datei werden vom Server eingelesen, sodass bei der Gestaltung des Textes keine Einschränkungen auf bestimmte Zeichen oder eine maximale Länge beachtet werden müssen. Stattdessen können beliebige HTML-Tags genutzt werden, um den Text frei zu gestalten. Die entsprechende Methode kann per GWT-RPC vom Client aufgerufen werden, wobei nur die Nummer übergeben werden muss, da die Sprache serverweit festgelegt und daher dort gespeichert ist.

6.1.5. Oberfläche

Die Texte des Tutorials werden in einer Dialogbox dargestellt, um sie von der Landschaft abzuheben und die Aufmerksamkeit des Benutzers zu erregen. Um dadurch die Bedienung von ExplorViz nicht zu behindern, darf die Dialogbox die Interaktion mit der Seite nicht verhindern oder Teile verdecken. Mit JQuery UI ist es leicht, eine Dialogbox zu erstellen, die vom Benutzer verschoben werden kann (siehe Listing 6.1). Der Dialog wird so angepasst, dass kein Icon zum Schließen angezeigt wird und er auch nicht mit der Escape-Taste geschlossen werden kann, damit der Benutzer das Tutorial nicht versehentlich schließen kann. Dadurch ist es nicht nötig, den Zustand des Dialogs zu überwachen und ihn wenn nötig neu zu öffnen.

Für die Schritte, in denen keine Aktion des Benutzers nötig ist, sondern nur Informationen vermittelt werden sollen, wird außerdem ein Button auf dem Dialog eingeblendet, mit dem das Tutorial fortgesetzt werden kann (siehe Listing 6.2).

Listing 6.1. Generierung der Dialogbox ohne Schließ-Möglichkeit

```
public static native void showTutorialDialog() /*-{
  $wnd.jQuery("#tutorialDialog").show();
  $wnd.jQuery("#tutorialDialog").dialog(
      {
        closeOnEscape : false,
        title : 'Tutorial',
```

```
          width : '500px',
          height : 'auto',
          open : function(event, ui) {
            $wnd.jQuery(this).closest('.ui-dialog').find(
                '.ui-dialog-titlebar-close').hide();
          },
          position : {
            my : 'center top',
            at : 'center center',
            of : $wnd
          }
        });
}-*/;
```

Listing 6.2. Hinzufügen eines Buttons zum bestehenden Dialog

```
public static native void showTutorialContinueButton() /*-{
  $wnd.jQuery("#tutorialDialog").dialog('option', 'buttons', {
    'Next >>' : function() {
      @explorviz.visualization.experiment.Experiment::incStep()()
    }
  });
}-*/;
```

Um dem Benutzer das Finden der richtigen Komponente zu erleichtern und Missverständnissen vorzubeugen, wird in den meisten Schritten neben der textuellen Anweisung auch ein Pfeil eingeblendet, der auf die Komponente deutet, mit der interagiert werden soll (siehe Abbildung 6.1.5). Ebenso wie beim Setzten der Handler wird dazu eine Überprüfung auf den Tutorialmodus in die `draw`- bzw `createDrawing`-Methoden für die verschiedenen Komponenten eingefügt. Handelt es sich bei der zu malenden Komponente um die, die im aktuellen Tutorialschritt benutzt wird, werden die Koordinaten der Komponente benutzt, um die Pfeilspitze zu platzieren. Der Pfeil selbst wird aus einem Dreieck und einem Viereck zusammengesetzt, sodass die schon existierenden primitiven Objekte genutzt werden können, anstatt ein neues Objekt zu erstellen. Diese beiden Komponenten werden anschließend in einer Liste von Polygonen gespeichert und erst zu der Liste der zu zeichnenden Polygone hinzugefügt, nachdem alle anderen Elemente hinzugefügt wurden. Dadurch wird sichergestellt, dass die Pfeil-Komponenten als letztes gezeichnet und nicht durch andere Komponenten verdeckt werden.

6.2. Fragebogenkonzept

Der folgende Abschnitt beschreibt die Implementierung des Fragebogenkonzepts in ExplorViz. Besonderes Augenmerk wird dabei auf die Integration in den normalen Betrieb gelegt,

Abbildung 6.1. Screenshot von ExplorViz im Tutorialmodus

die ähnlich wie beim Tutorialmodus zu großen Teilen durch Vererbung erreicht wird. Außerdem wird die Kommunikation mit dem Server und dessen Aufgabe der Datenhaltung beschrieben, sowie der Aufbau der Fragebogenoberfläche erklärt.

6.2.1. Ablauf

Der Ablauf eines Experimentes sieht vor, dass nach dem Einloggen zuerst das Tutorial abgeschlossen werden soll, um dem Teilnehmer die Funktionen von ExplorViz zu erklären. Daran anschließend sollen einige personenbezogene Daten abgefragt werden, anhand derer bestimmte Charakteristika der jeweiligen Gruppe festgestellt werden können. Danach wird der Fragebogen automatisch gestartet und die dafür vorgesehene Softwarelandschaft geladen. Dabei wird zusätzlich auf die Koordination von Fragen und Zeitpunkten geachtet. Daraufhin werden dem Benutzer weitere konfigurierbare, meinungsbasierte Fragen gestellt, nach deren Beantwortung er automatisch ausgeloggt wird.

Start

Ob der Fragebogenmodus genutzt werden soll oder nicht kann in den administrativen Einstellungen festgelegt werden. Diese Einstellung entscheidet, ob die Benutzer nach dem Abschluss des Tutorials ExplorViz frei nutzen können oder zu einer Versuchsgruppe gehören. Da die Einstellung auf dem Server festgelegt ist, muss dies nicht für jeden Versuchsteilnehmer einzeln festgelegt werden.

Ist das Tutorial beendet und die Einstellung für das Experiment gesetzt, wird vom `LandscapeExchangeManager` nicht die normale Landschaft angefordert, sondern die auf dem Server abgelegte, vorher aufgezeichnete Landschaft. Zusätzlich erscheint der Dialog, in der der Fragebogen angezeigt werden wird.

Personenbezogene Daten

Bevor die erste Frage des eigentlichen Fragebogens gestellt wird, werden statistische Informationen über den Versuchsteilnehmer erfragt. Diese Fragen werden im selben Dialog angezeigt und sind auf zwei einzeln angezeigte Formulare unterteilt. Zu diesen statistischen Informationen gehören Alter, Geschlecht, höchster Abschluss sowie der Erfahrungsgrad mit ExplorViz bzw. einem Vergleichstool, Erfahrung mit der Analyse von Programmen und dem zu untersuchenden Programm selbst. Die eingegebenen Daten werden mithilfe des required-Attributs der Formulare und *.validate* überprüft. Die Antworten werden anhand der Userkennung auf dem Server gespeichert (siehe Abschnitt 6.2.4). Erst wenn alle Fragen beantwortet wurden, wird die erste Frage geladen.

Kopplung mit Replayer

Wie in Unterabschnitt 5.2.4 erläutert, kann es notwendig sein, Fragen mit bestimmten Zeitpunkten der Visualisierung zu koppeln. Um dies zu erreichen werden zu den Fragen die Zeitstempel gespeichert, bis zu denen die Visualisierung abgespielt werden darf. Die Zeitstempel der Landschaften können dem Namen der aufgezeichneten Daten entnommen werden.

Während das Experiment läuft, wird anstelle des normalen `LandscapeExchanges` der `LandscapeReplayer` gestartet, der über ein Attribut gesteuert wird, dass den höchsten zu ladenden Zeitstempel angibt. Wird eine neue Frage geladen, wird dieser Wert auf den in der Frage gespeicherten Wert gesetzt.

Der `LandscapeReplayer` verhält sich normal, solange die zu ladenden Landschaften innerhalb des vorgegebenen Zeitraums liegen. Sobald er aber eine Landschaft laden müsste, die einen höheren Zeitstempel aufweist, hält der Replayer an und fährt erst fort, wenn ein neuer, höherer Zeitstempel übergeben wird.

Bearbeiten des Fragebogens

Die Fragen werden einzeln, nacheinander in einem Dialog angezeigt. Falls die entsprechende Einstellung gesetzt ist, können Fragen übersprungen und dadurch leere Antworten gegeben werden. Ist dies nicht der Fall, muss das Formular vollständig ausgefüllt werden, bevor fortgefahren werden kann. Vollständig bedeutet hierbei, dass jedes Inputfeld einen Inhalt haben muss, ein Radiobutton ausgewählt oder mindestens eine Checkbox aktiviert sein muss. Da die Daten sofort nach dem Absenden gespeichert werden (siehe Abschnitt 6.2.4), ist es nicht möglich, zu einer vorherigen Frage zurückzukehren.

Die Angabe, wie viel Zeit zur Bearbeitung der Frage vorgesehen ist, ist keine harte Vorgabe, das heißt es wird nicht automatisch zur nächsten Frage übergegangen. Sie dient stattdessen dazu, dem Benutzer einen Zeitrahmen vorzugeben. Die Entscheidung, eine leere bzw. absichtlich falsche Antwort zu geben, wenn er diesen Rahmen überschreitet, ist ihm daher selbst überlassen.

Abschließende Fragen

Im Anschluss an die Fragen zum Vergleich der Programme folgen bis zu drei Formulare zur Bewertung des Programms, des Tutorials und dem Schwierigkeitsgrad der gestellten Fragen. Diese dienen sowohl der möglichen Verbesserung des Programms als auch zur zusätzlichen Einschätzung der Ergebnisse, da neben der Korrektheit auch interessant ist, wie schwer die Fragen empfunden wurden.

6.2.2. Konfigurationsmöglichkeiten

Die Einstellung des Fragebogens ist, ob er überhaupt gestartet werden soll. Zusätzlich kann eingestellt werden, ob es möglich sein soll, Fragen zu überspringen, d.h. leere Antworten zu geben. Leere Antworten haben den Vorteil, dass leicht zu erkennen ist, bei welcher Aufgabe ein Benutzer aufgegeben hat, während es sonst beim Benutzer liegt, entsprechende Antworten erkennbar zu machen. Der Nachteil ist jedoch, dass es den Nutzer dafür verleiten kann, früher aufzugeben, und leere Antworten bei der Auswertung als Sonderfall behandelt werden müssten, diese also erschweren.

Außerdem wird die Spracheinstellung, die für das Tutorial implementiert wurde, auch vom Fragebogen genutzt. Sie hat keinen Einfluss auf die Fragen, da nur eine Datei für Fragen vorgesehen ist, sodass immer nur Fragen in einer Sprache existieren. Die Sprache wird stattdessen für die Ausgabe der Meldungen des Validierungsframeworks *.validate* genutzt.

6.2.3. Oberfläche

Die Oberfläche besteht aus einem HTML-Formular, das in einem JQuery UI Dialog angezeigt und mithilfe von JavaScript ausgelesen wird.

Aufbau des Formulars

Da drei verschiedene Fragetypen unterstützt werden, erfolgt beim Aufbau des Dialoginhaltes eine Fallunterscheidung über den Fragetyp. Dadurch wird entschieden, ob unter dem Fragetext Eingabefelder, Radiobuttons oder Checkboxes für jede Antwortmöglichkeit angezeigt werden (siehe Listing 6.3).

Für ein einheitliches und ansprechendes Aussehen des Formulars wird Bootstrap verwendet, das zudem Funktionen wie Tooltips und Popovers zur Verfügung stellt.

Listing 6.3. Auszug aus dem Fragedialog

```
var StringBuilder html = new StringBuilder()
html.append("<p>"+question.text+"</p>")
html.append("<form class='form' role='form' id='questionForm'>")
var String[]  ans = question.answers
html.append("<div class='form-group' id='form-group'>")
if(question.type.equals("Free")){
  html.append("<label for='input'>Answer</label>")
  html.append("<div id='input' class='input-group'>")
  if(question.freeAnswers > 1){
    for(var i = 0; i < question.freeAnswers; i++){
      html.append("<input type='text' class='form-control'
        id='input"+i.toString()+"' placeholder='Enter Answer'
        name='input"+i.toString()+"' minlength='1' autocomplete='off'
        required>")
     }
  }else{ //only one question gets a textbox
    html.append("<textarea class='form-control questionTextarea' id='input1'
      name='input1' rows='2' required></textarea>")
  }
  html.append("</div>")
}
```

Während zum Erfragen der personenbezogenen Daten und Anmerkungen überwiegend Comboboxen genutzt werden, um eine Auswahl von Antwortmöglichkeiten zu geben, werden MC-Fragen innerhalb des Experiments selbst durch Radiobuttons dargestellt, um alle Antwortmöglichkeiten gleichzeitig sichtbar zu machen.

Durch die Nutzung von Radiobuttons für einfache MC-Fragen und Checkboxen für mehrfache MC-Fragen wie es üblich ist, ist keine Erklärung der Symbolik notwendig, wie es der Fall wäre, wenn eigene Mittel zum Erfragen der Antworten genutzt worden wären.

Für die Bewertung des Tutorials und des Fragebogens wird eine Intervallskala genutzt, in dem Zahlen für die Ausprägung der Antwort gewählt werden können und angegeben ist, welches Ende der Zahlenreihe einer guten und welches einer schlechten Bewertung entspricht. Zahlen wurden als Repräsentation gewählt, um die Mehrsprachigkeit zu ver-

einfachen und nicht zusätzlich zu den übrigen Beschriftungen auch alle Ausprägungen im Vokabular hinterlegen zu müssen. Weiterhin wurde eine ungerade Anzahl von Antwortmöglichkeiten gewählt, um dem Benutzer bewusst die Möglichkeit einer neutralen Antwort zu geben. Durch diese Möglichkeit ist anzunehmen, dass nicht neutrale Antworten absichtlich so gewählt wurden. Fehlt hingegen eine neutrale Antwortmöglichkeit, würden als Ersatz die beiden mittleren Antwortmöglichkeiten gewählt werden und dadurch das Meinungsbild leicht verzerren.

Aufbau des Dialogs

Das abhängig von den Fragetypen erstellte Formular wird in einen JQuery UI Dialog eingebettet, der keinen Button zum Schließen bereitstellt und auch nicht mit Esc geschlossen werden kann (siehe Abbildung 6.2). Ebenso ist die Größe nicht veränderbar, dafür kann der Dialog per drag und drop frei bewegt werden. Unterhalb des Formulars wird ein Ok-Button eingeblendet, der für das Absenden des Formulars zuständig ist.

Beim Klick auf den Ok-Button wird *.validate* genutzt, um leere Eingaben abzufangen, sodass es versehentliches Absenden leerer Antworten unterbunden wird und nur über den Skip-Button erfolgen kann. Das Drücken der Enter-Taste wird dabei wie ein Klick auf den Ok-Button interpretiert. Diese Funktionalitäten werden mit Hilfe von JQuery erreicht, indem Eingaben der Tastatur abgefangen und die Aktionen der Buttons manipuliert werden. Auch die Einstellungen des Validators werden beim Aufbau der entsprechenden Dialogbox vorgenommen.

Um den Benutzer nicht dazu zu motivieren, beliebige Antworten zu geben, wenn er eine Frage nicht beantworten kann, ist neben dem Button zum Absenden der Antwort auch ein Button zum Überspringen der Frage vorhanden, der möglicherweise schon getätigte Eingaben verwirft und eine leere Antwort erstellt. Ob dieser Button angezeigt wird oder nicht kann in den administrativen Optionen für den Server festgelegt werden (siehe Unterabschnitt 6.2.2).

Die Titelzeile des Dialogs wird zur Fortschrittsanzeige genutzt, indem die Nummer der aktuellen Frage und die Anzahl der insgesamt vorhandenen Fragen angezeigt werden. Fortschrittsanzeigen können bei sehr langen Fragebögen einen negativen Effekt haben, da sie dem Benutzer zeigen, dass er noch viele Fragen beantworten muss. Bei kürzeren Fragebögen hingegen sind sie motivierend, weil das Ziel schon absehbar ist. Da das Experiment nur einen begrenzten Fragenumfang haben wird, fällt es in die zweite Kategorie.

Unterhalb der Eingabemöglichkeit für die Antwort wird ein Timer eingeblendet, der anzeigt, wie lange die Frage schon bearbeitet wird. Wird ein Wert überschritten, der je nach Frage festgelegt werden kann, wird die Anzeige rot und um den Hinweis *overtime* ergänzt. Der Benutzer soll dadurch darauf hingewiesen werden, dass er die Frage bald beantworten oder zur nächsten Frage übergehen sollte, wenn er nicht das Gefühl hat, der Antwort nahe zu sein.

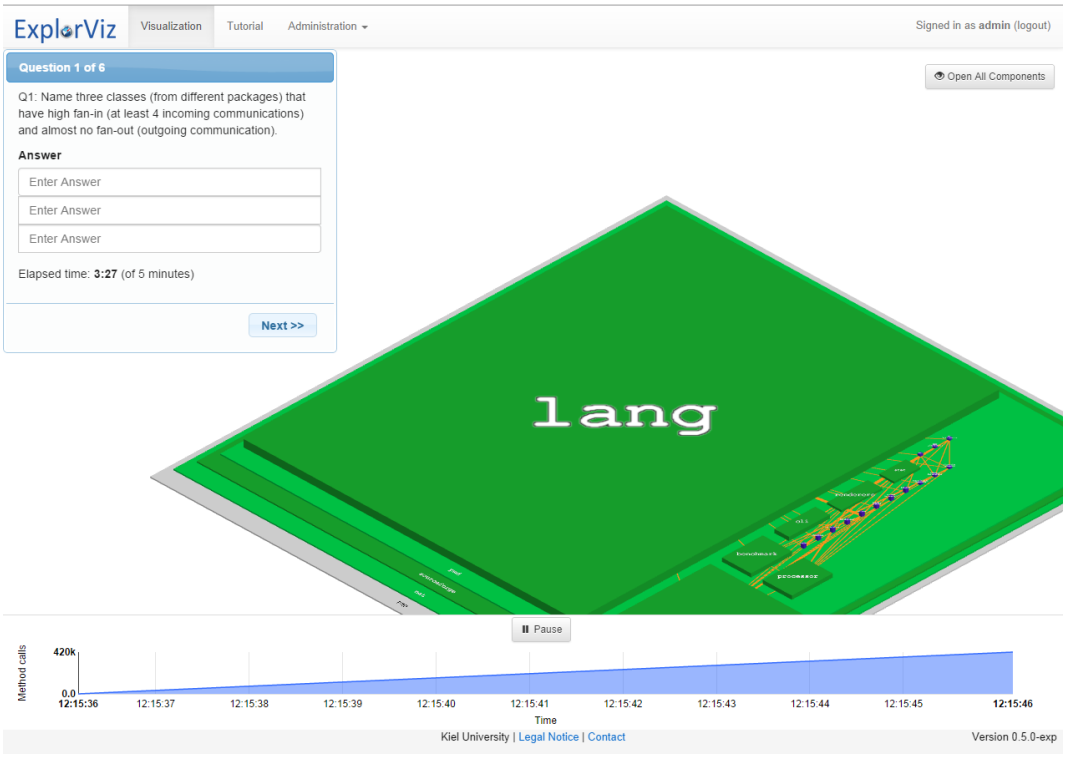

Abbildung 6.2. Screenshot von ExplorViz mit eingeblendeter Frage

6.2.4. Datenhaltung

Für die Durchführung eines Experiments müssen verschiedene Daten eingelesen und gespeichert werden. Dazu gehört zum einen das Vokabular, dass für das Erfragen von personenbezogenen Daten und Anmerkungen genutzt wird, als auch insbesondere die Verwaltung der Fragen und Antworten.

Da die Fragen auf dem Server abgelegt sind und auch die Antworten dort gespeichert werden sollen, werden diese Daten per GWT-RPC zwischen Client und Server ausgetauscht.

Einlesen von Fragen

Zu Beginn des Experiments wird eine Anfrage an den Server gesendet, der daraufhin die gegebenen Fragen-Datei einließt und entsprechend des Frageschemas parst, siehe Listing 6.4. Die einzelnen Fragen werden in einer dafür vorgesehenen Klasse gespeichert und in dieser Form als Array an den Client gesendet. Dort werden sie in einer statischen Variable abgelegt, sodass sie jederzeit ohne weitere Kommunikation mit dem Server zur Verfügung stehen.

Da jeweils nur die Fragen für ein Experiment hinterlegt werden können, muss die Mehrsprachigkeit hierbei nicht beachtet werden, sondern ergibt sich durch die Sprache, in der die Fragen gestellt werden. Es wird daher immer die gesamte Datei *questions.txt* eingelesen.

Listing 6.4. Einlesen der Fragen

```
final String filePath = getServletContext().getRealPath("/experiment/")
  + "/questions.txt";
String text, answers, corrects, time, free;
final BufferedReader br = new BufferedReader(new FileReader(filePath));
text = br.readLine(); // read text
int i = 0;
while (null != text) {
  answers = br.readLine(); // read answers
  corrects = br.readLine(); // read correct answers
  free = br.readLine(); // read amount of free inputs
  time = br.readLine(); // read timestamp
  questions.add(new Question(i, text, answers, corrects, free, time));
  text = br.readLine(); // read text of next question
  i++;
}
br.close();
```

Für die einführenden und abschließenden Fragen wird ebenfalls eine Datei eingelesen und anhand der Keywords am Zeilenbeginn geparst. Das ist notwendig, da je nach gewünschter Eingabemöglichkeit unterschiedliche Attribute zu Konfiguration gesetzt werden können, und daher unterschiedlich viele Zeilen zu einer Frage gehören. Wurde eine Frage vollständig eingelesen, wird sie abhängig davon, in welchem Teil der Datei sie stand, in einer Liste abgelegt, sodass für jedes zu erstellende Formular eine solche Liste aller dazugehörigen ragen vorhanden ist.

Speichern von Antworten

Clientseitig werden die eingegebenen Daten per Javascript ausgelesen und nach dem Parsen in Antwort-Objekten abgelegt, die anschließend per RPC an den Server übermittelt werden. Dadurch, dass nicht sofort ein String übergeben wird, ist es leichter, Änderungen am Speicherformat vorzunehmen oder andere Formate zu ergänzen. Sollen andere Formate genutzt werden, müssen daher nur neue Methoden implementiert werden, ohne in den Datenfluss einzugreifen. Dies gilt genauso für die Änderung der bestehenden Methode.

Nach der Konvertierung in das gewünschte CSV-Format auf dem Server werden die Antworten durch einen FileWriter an das Ende einer durch die ID des Users gekennzeichnete Datei geschrieben (siehe Listing 6.5). Die Übertragung erfolgt nach jeder Frage, damit im Fall eines Programmabsturzes oder Abbruches des Benutzers möglichst viele Daten

gesammelt werden können. Die Aufteilung der Daten in Bezug auf den Benutzer verhindert, dass durch Nebenläufigkeit ein Benutzer die Antwort eines anderen überschreiben kann.

Aufgrund der Art, wie Javascript den Inhalt des Formulars übergibt, können nur bis zu zehn Freitextfelder korrekt geparst werden. Werden mehr Textfelder genutzt, kann der korrekte Substring nicht ermittelt werden, da dieser von der Länge des Namens des Feldes abhängt. Nachträgliche Versuche, überzählige Zeichen abzuschneiden, würden den Zeichensatz beschränken, der für die Beantwortung der Fragen genutzt werden kann.

Listing 6.5. Schreiben der Antwort

```
public void writeAnswer(final Answer answer) throws IOException {
  String id = answer.getUserID();
  if (id.equals("")) {
    id = "DummyUser";
  }
  writeString(answer.toCSV(), id);
}
public void writeString(final String string, final String id)
throws IOException {
  if (folder == null) {
    folder = FileSystemHelper.getExplorVizDirectory() + "/" + "experiment";
    new File(folder).mkdir();
  }
  try {
    answerFile = new FileOutputStream(new File(
      FileSystemHelper.getExplorVizDirectory() + "/experiment/"
      + "/" + id + ".csv"), true);
    final String writeString = id + "," + string;
    answerFile.write(writeString.getBytes("UTF-8"));
    answerFile.flush();
  } catch (final FileNotFoundException e) {
    Logging.log(e.getMessage());
  }
}
```

GUI zur Eingabe von Fragen

Im Administrationsmenü wurde ein Eintrag hinzugefügt, der ein Formular zum Erstellen neuer Fragen aufruft, siehe Abbildung 6.3. Die in das Formular eingegebenen Daten werden so formatiert, dass sie in dem in Unterabschnitt 5.2.2 vorgestellten Format in die Fragen-Datei geschrieben werden. Es kann nur jeweils eine Frage zur Zeit eingegeben werden und es stehen zwei Speichermöglichkeiten zur Verfügung.

Die erste Möglichkeit ist zum Anlegen eines neuen Fragebogens gedacht und überschreibt mit der neuen Frage alle bisher gespeicherten Fragen. Die zweite Möglichkeit fügt die neue Frage an das Ende der bisherigen Fragen an.

Dies bedeutet insbesondere, dass bestehende Fragen nicht editiert werden können. Stattdessen müssen die Fragen gelöscht und mit den Änderungen erneut eingetragen werden, wenn mit der Oberfläche gearbeitet wird. Bestehen Zugangsrechte zum Server ist es für kleine Änderungen daher leichter, die Fragen-Datei direkt zu bearbeiten.

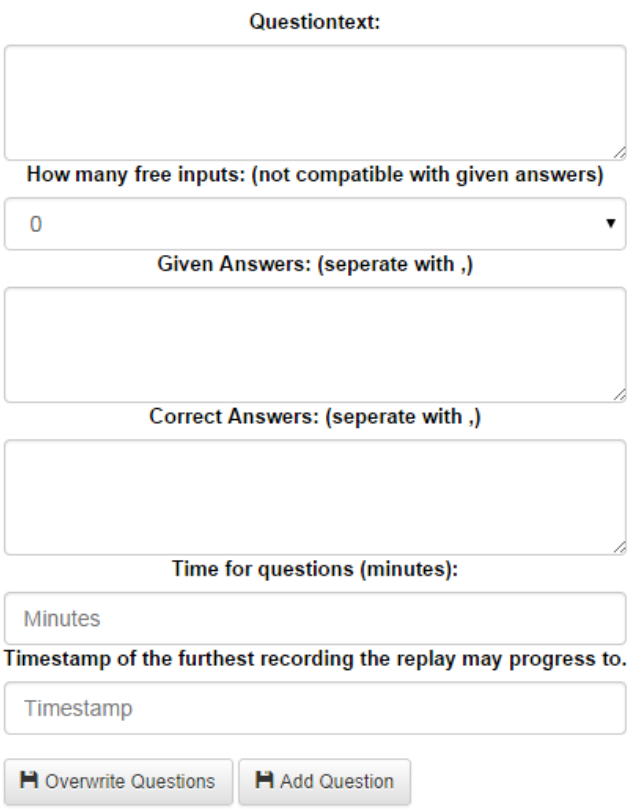

Abbildung 6.3. Screenshot der Eingabeoberfläche für Fragen

Menü zum Export der gegebenen Antworten

Um die Antworten auf die Fragen nicht von Hand vom Server kopieren zu müssen, wozu Zugangsrechte auf dem Server nötig sind, wurde im Administrationsmenü ein

Menüpunkt eingefügt, der alle Antworten in ein Zip-Archiv verpackt. Diese Datei wird als byte-Array eingelesen und mit Base64 codiert an den Client geschickt, wo sie mithilfe von Javascript decodiert und gespeichert werden. Wichtig ist dabei das Einlesen als byte-Array im Gegensatz zum String, da die gespeicherte Datei andernfalls zwar mit der Endung .zip gespeichert werden kann, sich aber nicht mehr als Archiv öffnen lässt.

Kapitel 7

Kontrollierte Experimente

Die beiden implementierten Funktionen (Tutorial und Fragebogen) sind in zwei Experimenten [Fittkau u. a. 2015] zum Einsatz gekommen, deren Aufbau und Ergebnisse in diesem Kapitel beschrieben werden.

Beide Experimente vergleichen ExplorViz mit dem Visualisierungstool Extravis[1] und unterscheiden sich durch die Größe des untersuchten Programms und der auf das Programm abgestimmten Fragen. Da Cornelissen [2009] eine Studie zum Vergleich von Extravis gegenüber Eclipse durchgeführt hat, wurde ein analoges Design für die Experiment gewählt. Dadurch wird verhindert, dass ein Experimentaufbau gewählt wird, der ExplorViz bevorzugt.

7.1. Experimentaufbau

Um ein Experiment durchzuführen, müssen Entscheidungen zu unterschiedlichen Bereichen getroffen werden. Diese Entscheidungen sind es, die das Experiment definieren und auch Grundlage der hinterher erfolgenden Analyse sind.

7.1.1. Zu untersuchende Fragen und Hypothesen

Der Rahmen der Studie wird durch drei Fragen gegebe, deren Beantwortung das Ziel der Studie darstellen:

- In welchem Verhältnis stehen die Bearbeitungszeiten mit Extravis bzw. ExplorViz zur Beantwortung von Programmverständnisfragen?
- In welchem Verhältnis steht die Korrektheit der Antworten auf Programmverständnisfragen bei Nutzung von Extravis zu der bei Nutzung von ExplorViz?
- Welche Programmverständnisfragen können mit der kreisförmigen Ansicht und welche mit der Stadtansicht besser bearbeitet werden?

Es wurden weiterhin zwei Nullhypothesen sowie zwei alternative Hypothesen formuliert, anhand derer ExplorViz und Extravis verglichen werden:

[1] http://www.win.tue.nl/ dholten/extravis/

- $H1_0$: Die Zeit, die zum Bearbeiten von Verständnisaufgaben mit Extravis und ExplorViz benötigt wird, unterscheidet sich nicht.
- $H2_0$: Die Korrektheit der gegebenen Antworten bei Nutzung von Extravis oder ExplorViz unterscheidet sich nicht
- $H1$: Bei der Bearbeitung von Aufgaben mit Extravis oder ExplorViz wird unterschiedlich viel Zeit benötigt.
- $H2$: Die Nutzung von Extravis oder ExplorViz wirkt sich auf die Korrektheit der gegebenen Antworten aus.

7.1.2. Abhängige und unabhängige Variablen

Bei der unabhängigen Variable handelt es sich um das Tool, das zur Bearbeitung der Programmverständnis-Aufgaben genutzt wird, also entweder um Extravis oder ExplorViz. Es werden keine zusätzlichen Tools genutzt, damit die Ergebnisse allein durch den Unterschied zwischen Extravis und ExplorViz begründet sind. Die abhängigen Variablen sind die Korrektheit der Antworten und die Zeit, die für die Antwort benötigt wird. Diese beiden Variablen werden überlicherweise in Studien zum Programmverständnis genutzt (siehe Kapitel 9).

7.1.3. Aufgaben

Da ein Vergleich mit Extravis durchgeführt wird, wurden Aufgaben gewählt, die denen von Cornelissen [2009] entsprechen. Dadurch wurde automatisch das Framework von Pacione u. a. [2004] berücksichtigt, das verschiedene Kategorien definiert, in die sich Aufgaben zum Programmverständnis einordnen lassen. Durch Auswahl von Aufgaben, die verschiedenen Kategorien zugeordnet werden können, wird sichergestellt, dass verschiedene Aspekte des Programmverständnis berücksichtigt werden.

Das Programm für das zweite Experiment wurde durch äußere Umstände vorgegeben. Die Teilnehmer des Softwareprojekts hatten die Aufgabe, das Programm Babsi[2] zu überarbeiten und zu verbessern, weshalb sich anbot, eines der Experimente mit Babsi und eben diesen Studenten durchzuführen. Da es sich bei Babsi um ein kleines Programm handelt, wurde für das andere, zuerst durchzuführende Experiment ein großes Programm ausgewählt.

Es wurde in Betracht gezogen, eine Ausführung von Checkstyle[3] analysieren zu lassen, die der von [Cornelissen 2009] entspricht. Dies wurde jedoch verworfen, da nur eine geringe Anzahl von Klassen genutzt werden würde, was dem Wunsch nach einem großen Tool als Gegensatz zu Babsi widersprach. Die Wahl fiel schließlich auf PMD[4], da die ähnliche

[2] http://babsi.sourceforge.net
[3] http://checkstyle.sourceforge.net/
[4] http://pmd.sourceforge.net

Funktionsweise von PMD und Checkstyle es ermöglichte, beinahe identische Aufgaben zu denen von [Cornelissen 2009] aufzustellen. Aus Zeitgründen wurden allerdings nur sechs Aufgaben ausgewählt, da pro Teilnehmer nur eine Stunde für das gesamte Experiment inklusive der Einweisung in das entsprechende Tool zur Verfügung stand. Dabei wurde darauf geachtet, solche Aufgaben auszuschließen, die nach den Kategorien von Pacione u. a. [2004] gleiche Bereiche des Programmverständnis abdecken, sodass keiner der Bereiche, die von Cornelissen [2009] berücksichtig wurden, ausgeschlossen wurde. Die Aufgabenstellungen, ihr Kontext und die Einordnung in die Kategorien von Pacione u. a. [2004] können Tabelle 7.1 und Tabelle 7.2 entnommen werden.

Zusätzlich wurden die Aufgaben als Freitextfragen gestellt, das heißt es wurden keine Antworten vorgegeben. Die Entscheidung gegen Multiple-Choice-Fragen wurde bewusst getroffen, um zu verhindern, dass die Ergebnisse durch geratene Antworten verfälscht werden. Zusätzlich wurden nur bei einer Aufgabe, bei der nach drei Klassen gefragt wurde, drei Inputfelder für die Antwort bereitgestellt, während alle anderen Aufgaben Textfelder enthielten. Wären bei der Frage nach den Methoden zwischen zwei Klassen mehr oder weniger Felder gegeben, als der Proband Methoden gefunden hätte, hätte ihn dies darauf hingewiesen, dass seine Antwort nicht richtig sein kann.

7.1.4. Versuchsteilnehmer

Bei den Versuchsteilnehmern handelte es sich um Studierende der Universität Kiel oder der Fachhochschule Kiel, die freiwillig (PMD-Experiment) oder im Zuge der Lehrveranstaltung *Softwareprojekt* (Babsi-Experiment) an dem Experiment teilgenommen haben. Zur Motivation der freiwilligen Probanden wurden Preise für die besten - das heißt korrektesten und zugleich schnellsten - Antworten ausgeschrieben. Da die Probanden des Babsi-Experiments im Laufe der Lehrveranstaltung Babsi verbessern sollen, haben sie im Gegensatz zu den freiwilligen Teilnehmern ein eigenes Interesse daran, das zu untersuchende Programm zu verstehen.

Da die beiden Programme von verschiedenen Gruppen von Versuchsteilnehmern analysiert werden sollen, wird ein *between subjects* Versuchsaufbau gewählt. Das bedeutet, dass jeder Versuchsteilnehmer entweder mit ExplorViz oder Extravis arbeitet, da das gleiche Programm nicht zweimal von der gleichen Person ohne Folgeeffekte untersucht werden kann. Die Einteilung der Versuchsteilnehmer auf die Kontroll- und Experimentgruppen erfolgte dabei zufällig.

Vor dem Beginn des eigentlich Fragebogens wurden die Teilnehmer außerdem befragt, wie viel Erfahrung sie mit objektorientierter Entwicklung, dynamischen Analysen, dem analysierten Programm (PMD oder Babsi) und dem zu nutzenden Tool (ExplorViz oder Extravis) haben.

Tabelle 7.1. Programmverständnis-Aufgaben des ersten Experiments (PMD)

ID	Kategorie	Aufgabenstellung	Punkte
		Kontext: Identifizieren von Refactoring-Möglichkeiten	
T1	A{4,8}	Nenne drei Klassen (aus verschiedenen Paketen) die hohen Fan-In (mindestens 4 eingehende Kommunikationen) und fast kein Fan-Out (ausgehende Kommunikationen) haben.	3
		Kontext: Verständnis für den Checking-Prozess gewinnen	
T2.1	A{3,4,5}	Nenne alle Konstruktor- und Methodenaufrufe zwischen `RuleChain` und `JavaRuleChainVisitor`.	3
T2.2	A{1,2,5,6}	Beschreibe den Lebenszyklus der Klasse `GodeClassRule`: Wer erzeugt sie, was tut sie (auf hohem Abstraktionslevel)?	3
		Kontext: Verständnis für den Reporting-Prozess gewinnen	
T3.1	{1,5}	Welche Regeln des gegebenen Regelsatzes werden von der Eingabedatei verletzt? Tipp: Da es sich um eine dynamische Analyse handelt, wird das Violation-Objekt nur für diese Fälle erzeugt.	2
T3.2	{1,3}	Wie funktioniert die Berichterstattung von Regelverletzungen? Von wo stammen die Regelverletzungen und wie werden sie dem Nutzer mitgeteilt. Nenne die Klassen, die direkt in den Prozess involviert sind. Tipp: Das Ausgabeformat ist HTML.	4
		Kontext: Allgemeines Verständnis gewinnen	
T4	A{1,7,9}	Ausgehend von der Mainclass `PMD` - Auf hohem Abstraktionslevel, was sind die Hauptschritte, die während einer Ausführung von PMD ausgeführt werden. Nenne nicht mehr als fünf Schritte. Tipp: Dies ist eine explorative Aufgabe, um einen Überblick über das System zu bekommen. Eine Strategie ist es, den Kommunikationen zwischen Klassen und Paketen zu folgen. Bedenke auch den Informationszettel zu PMD.	5

Tabelle 7.2. Programmverständnis-Aufgaben des zweiten Experiments (Babsi)

ID	Kategorie	Aufgabenstellung	Punkte
		Kontext: Identifizieren von Refactoring-Möglichkeiten	
T1	A{4,8}	Nenne drei Klassen die hohen Fan-In (mindestens 3 eingehende Kommunikationen) und fast kein Fan-Out (ausgehende Kommunikationen) haben.	3
		Kontext: Verständnis für den Login-Prozess gewinnen	
T2.1	A{3,4,5}	Nenne alle Konstruktor- und Methodenaufrufe zwischen `gui.MainActivity` und `comm.Sync`.	3
T2.2	A{1,2,5,6}	Beschreibe den Lebenszyklus der Klasse `data.User`: Wer erzeugt sie, wie wird sie genutzt? Schreibe die Methodenaufrufe auf.	3
		Kontext: Verständnis für die Antibiotika-Anzeigen gewinnen	
T3	A{1,3}	Wie funktioniert das Anzeigen von Antibiotika? Wo und wie werden sie erstellt? Schreibe alle Klassen auf, die direkt in den Prozess involviert sind.	6
		Kontext: Allgemeines Verständnis gewinnen	
T4	A{1,7,9}	Ausgehend von der Mainclass `gui.MainActivity` - Welche Nutzeraktionen (z.B. Login und Logout) wurden während der aufgezeichneten Ausführung von Babsi ausgeführt. Schreibe die Klassen der Aktivitäten/Fragemente für jede Nutzeraktion auf. Nenne nicht mehr als sieben Schritte (Login und Logout ausgenommen). Tipp: Dies ist eine explorative Aufgabe, um einen Überblick über das System zu bekommen. Eine Strategie ist es, den Kommunikationen zwischen Klassen und Paketen zu folgen.	7

7.1.5. Deaktivierte Funktionen von ExtraViz

Da ExtraViz über einige Funktionen verfügt, die Extravis nicht besitzt, wurden diese zur besseren Vergleichbarkeit der Visualisierungen deaktiviert. Dies betrifft den Codeviewer und den Trace Replayer, da Extravis keinen internen Codeviewer zur Verfügung stellt und nur einen Trace zur Zeit anzeigen kann. Die Timeshift-Funktion wurde nicht explizit deaktiviert, jedoch nicht genutzt, da nur Daten für einen Zeitpunkt gegeben waren.

7.2. Ausführung

Im Folgenden wird beschrieben, wie die Daten für die Experimente gewonnen wurden, welche Tutorials zur Verfügung gestellt wurden und welchen Nutzen der elektronische Fragebogen erfüllen sollte. Weiterhin werden die Auswirkungen der Pilotstudie vorgestellt sowie der Ablauf der Experimente selbst beschrieben.

7.2.1. Daten für Extravis und ExplorViz

Da kein Tool gefunden wurde, mit dem aus einer Programmausführung das für Extravis nötige Eingabeformat gewonnen werden konnte, wurden zuerst die Daten für ExplorViz erzeugt und anschließend durch einen extra für das Experiment hinzugefügten Exporter in das von Extravis benötigte Rigi Standard Format [*Rigi user's manual - version 5.4.4*] übertragen.

7.2.2. Tutorials

Sowohl für ExplorViz als auch für Extravis wurden den Probanden Tutorials zur Verfügung gestellt. Im Fall von ExplorViz handelt es sich dabei um eine Form des im Zuge dieser Arbeit implementierten Tutorials. Es musste angepasst und gekürzt werden, da einige Funktionen von ExplorViz für die Dauer des Experiments deaktiviert wurden und die Probanden nicht durch die Erklärung von Funktionen, die sie gar nicht nutzen können, verwirrt werden sollten.

Für Extravis konnte kein interaktives Tutorial zur Verfügung gestellt werden. Stattdessen wurden bebilderte Erklärungen der Funktionen gegeben, die am Trace eines vom Experiment unabhängigen Programms ausprobiert werden konnten. Wie ausführlich dies geschah, blieb allerdings den Probanden überlassen.

7.2.3. Fragebogen

Anstelle eines gedruckten Fragebogens wurde das im Zuge dieser Arbeit implementierte Fragebogensystem für ExplorViz genutzt. Durch kleine Änderungen war es möglich, den Fragebogen auch ohne die normale ExplorViz-Umgebung im Browser anzeigen zu lassen, sodass er auch von der Kontrollgruppe genutzt werden konnte. Ein solcher automatischer Fragebogen bietet einige Vorteile gegenüber einem Fragebogen auf Papier. Zum einen kann verhindert werden, dass die Probanden zu schon bearbeiteten Aufgaben zurück kehren oder zuerst alle Aufgaben lesen, bevor sie beginnen. Beides würde es schwer machen, die Zeit zu messen, die für die Bearbeitung einer konkreten Aufgabe benötigt wurde. Ein weiterer Vorteil besteht in eben dieser Zeitmessung, da der Fragebogen zusätzlich zu der Antwort auch die Zeit speichert, die der Proband für die Aufgabe benötigt hat. Diese berechnet sich dabei aus der Differenz zwischen dem Absenden der vorherigen Aufgabe beziehungsweise des Beginns des Fragebogens und dem Absenden der aktuellen Antwort.

Des Weiteren ermöglicht es das Fragebogensystem, ein Fortschreiten ohne gegebene Antwort zu verhindern, sodass insbesondere sicher gestellt wird, dass die Probanden die Fragen zu ihrer Person und die abschließenden Fragen zum Experiment beantworten. Bei Antworten auf Zetteln könnte nur verhindert werden, dass Felder leer gelassen werden, indem die Abgaben noch im Beisein des Probanden überprüft und fehlende Informationen nachgetragen werden. Daraus würde zusätzlicher Aufwand entstehen, der den Ablauf des Experiments aufhalten würde.

7.2.4. Pilotstudie

Vor der Durchführung der Experimente wurde jeweils eine Pilotstudie durchgeführt. Diese diente dem Zweck, herauszufinden, ob die vorgesehene Bearbeitungszeit für die einzelnen Aufgaben realistisch ist, und ob die Aufgaben verständlich formuliert wurden. Die Aufgaben, bei denen Klärungsbedarf bestand, wurden entsprechend überarbeitet, um Missverständnissen bei der Bearbeitung entgegen zu wirken. Es wurde außerdem die Reihenfolge der Fragen verändert, um einen aufsteigenden Schwierigkeitsgrad zu erreichen. Dadurch dienen die ersten, leichteren Aufgaben zugleich als Einarbeitung in das jeweilige Tool, anstatt den Probanden direkt mit einer komplexeren Aufgabe zu konfrontieren.

7.2.5. Durchführung

Zu Beginn des Experiments, noch vor dem Start des jeweiligen Tutorials haben die Probanden eine kurze schriftliche Erklärung von PMD beziehungsweise Babsi erhalten, in der die grundlegende Aufgabe des Programms beschrieben und möglicherweise unbekannte Wörter und Abkürzungen erläutert wurden, die als Paket- oder Klassennamen auftauchen. Anschließend wurden die Probanden dazu aufgefordert, Fragen zu stellen, falls sie etwas nicht verstehen, bevor ihnen die Daten gegeben wurden, mit denen sie sich einloggen konnten. Daraufhin begann das entsprechende Tutorial, woraufhin automatisch der Fragebogen folgte, der sich aus den personenbezogenen Daten einschließlich der Erfahrungseinschätzung, den Aufgaben selbst sowie den abschließenden Fragen zum Tool und Experiment zusammensetzt.

Für die einfacheren Aufgaben wurden 5 Minuten angesetzt, während die komplexeren Aufgaben innerhalb von 10 Minuten bearbeitet werden sollten. Sowohl die vorgegebene Zeit als auch die schon verstrichene Bearbeitungszeit wurden den Probanden angezeigt. Sie wurden gebeten, sich an die Zeitvorgaben zu halten, jedoch nicht durch das Fragebogensystem dazu gezwungen.

7.3. Gesammelte Daten

Im Zuge der Experimente wurden nicht nur die gegebenen Antworten und die benötigte Zeit gespeichert, sondern auch die Strategien mithilfe von Videoaufzeichnungen, die die Probanden zum Lösen der Aufgaben genutzt haben. Zusätzlich wurden im Anschluss an die Aufgaben Anmerkungen zu den verschiedenen Bestandteilen des Experiments erbeten.

7.3.1. Zeitmessung und Userverhalten

Die Messung der Bearbeitungszeit der einzelnen Aufgaben erfolgte automatisch durch den elektronischen Fragebogen vorgenommen. Zusätzlich wurde ein Screen Capture Programm genutzt, um aufzuzeichnen, wie die Probanden die Aufgaben bearbeitet haben.

Diese Aufnahmen können genutzt werden, um das Vorgehen der Probanden zu untersuchen. Außerdem war es durch diese Aufnahmen möglich, bei technischen Problemen die Zeitmessung manuell zu korrigieren, und Gründe für auffällig schlechte Antworten herauszufinden (siehe Unterabschnitt 7.4.2).

7.3.2. Korrektheit

Zu jeder Frage wurde neben der gegebenen Antwort die Zeit gespeichert, die der Benutzer zur Bearbeitung benötigt hat. Die Messung beginnt, wenn die Frage angezeigt wird, und endet, sobald der Teilnehmer auf den Ok-Button klickt. Die Antworten werden, bis auf Konvertierung von Sonderzeichen, ohne Veränderung gespeichert.

Zum Benutzer wird neben dem Alter, höchsten Abschluss und momentanem Studienstand insbesondere der Erfahrungsgrad mit dem zu bedienendem und untersuchendem Tool sowieso objektorientierter Programmierung und dynamischen Analysen abgefragt. Für diese wurde die gleiche Skala benutzt wie von Cornelissen [2009], die fünf Skalenpunkte von keiner Erfahrung bis Experte umfasst.

Anschließend an das Experiment wurde erfragt, wie schwer die Aufgaben empfunden wurden und ob der Nutzer Zeitdruck verspürt hat. Weiterhin wurde um Anmerkungen zum Tutorial und dem genutzten Tool selbst gebeten.

7.3.3. Feedback

Anschließend an die Aufgaben wurden die Probanden gebeten, sowohl das genutzte Tool als auch das Tutorial zu bewerten und Verbesserungsvorschläge zu machen. Genannten Probleme im Zusammenhang mit ExplorViz waren die Überlagerung von Klassennamen sowie von Kommunikationslinien, die es erschweren, über einer bestimmten Kommunikation zu hovern. Außerdem wurde angegeben, dass ein Dialog anstelle eines Popovers zum Anzeigen der Methoden in einer Kommunikationslinie bevorzugt werden würde. Bei Extravis wurde die Möglichkeit vermisst, Kommunikationen, die nicht zu einer markierten Klasse gehören vollständig ausblenden zu können. Außerdem wurde die Ergänzung einer Suchfunktion vorgeschlagen und kritisiert, dass das Anzeigen der Methodenaufrufe sehr langsam ist. Zur Bewertung des Tutorials siehe Kapitel 8.

Weiterhin wurde eine Bewertung des Fragebogens erzielt, indem die Probanden die Schwierigkeit der einzelnen Aufgaben bewerten sollten und angeben sollten, ob sie Zeitdruck verspürt haben.

7.4. Ergebnisse

Im Folgenden wird beschrieben, welche Ergebnisse in Bezug auf Zeit und Korrektheit aus den beiden Experimenten gewonnen werden konnten. Dabei wurde die Probanden mit weniger als insgesamt drei Punkten wurden von der Analyse der Ergebnisse ausgeschlossen.

Dies betraf insgesamt sechs Probanden, fünf aus dem ersten und einen aus dem zweiten Experiment. In drei Fällen hat der Proband das zu untersuchende Programm gar nicht geöffnet, zwei weitere Probanden haben es versäumt, den gesamten Trace in Extravis anzeigen zu lassen, sodass sie nur 0,02% des Traces als Grundlage für ihre Antworten hatten, und der sechste Proband hat der Aufzeichnung seines Verhaltens nach überhaupt nicht versucht, die Aufgaben zu bearbeiten. Bei drei weiteren Probanden traten derartige Probleme während eines Teils der Aufgaben auf, weshalb für die entsprechenden Aufgaben keine gültigen Werte vorliegen. Sie wurden daher ebenfalls aus der Gesamtauswertung ausgeschlossen, jedoch bei der Analyse der einzelnen Aufgaben berücksichtigt. Eine Übersicht der Ergebnisse kann Tabelle 7.3 entnommen werden.

Um die Signifikanz der Resultate zu bestimmen, wurden zweiseitige Student's t-Tests durchgeführt, die gegen den Wert $\alpha = 0,05$ als Signifikanzgrenze verglichen wurden. Der Test setzt dabei Normalverteilung und gleiche Varianz der Daten der beiden Gruppen voraus. Um die Normalverteilung zu überprüfen, wurde der Shapiro-Wilk-Test [Shapiro und Wilk 1965] genutzt und zur Überprüfung der Varianz der Levene-Test [Levene 1960].

7.4.1. Benötigte Zeit

Die erste Frage, die durch die Experimente beantwortet werden sollte, war die Frage danach, in welchem Verhältnis die Zeiten stehen, die Probanden bei der Nutzung von Extravis oder ExplorViz benötigen. Die dazu gehörende Nullhypothese $H1_0$ besagt, dass kein Unterschied besteht.

Aus Tabelle 7.3 kann entnommen werden, dass die Benutzer von ExplorViz im ersten Experiment 28,06% und im zweiten Experiment 7,64% weniger Zeit benötigten. Sowohl der Shapiro-Wilk-Test als auch der Levene-Test fielen für beide Experimente positiv aus, sodass der t-Test durchgeführt werden konnte. Für das erste Experiment wurde dabei ein Ergebnis von 0,0002 erzielt, das unterhalb der gewählten Signifikanzgrenze von 0,05 liegt, weshalb die Nullhypothese $H1_0$ zugunsten der alternativen Hypothese $H1$ zurückgewiesen werden konnte.

Der t-Test zur Bestimmung der Signifikanz des zweiten Experiments hatte ein Ergebnis von 0,2362, was über der gewählten Grenze liegt. Daher kann die Nullhypothese für das zweite Experiment nicht zurückgewiesen werden.

Ohne weitere Experimente kann nicht festgestellt werden, welche Eigenschaften der visualisierten Programme (zum Beispiel Größe oder Struktur) für die unterschiedlichen Ergebnisse der beiden Experimente verantwortlich sind. Daher ist es nicht möglich, eine allgemeine Aussage über die benötigte Zeit zu treffen.

7.4.2. Korrektheit

Bei der zweiten Frage ging es darum, herauszufinden, ob sie die Korrektheit der Antworten unterscheidet, wenn zur Bearbeitung Extravis oder ExtraViz genutzt wird. Die Nullhypothese $H2_0$ besagt, dass kein Unterschied in der Korrektheit vorliegt.

Tabelle 7.3. Darstellung der benötigten Zeit (in Minuten) und Korrektheit (in Punkten)

	Benötigte Zeit		Korrektheit	
	Extravis	ExplorViz	Extravis	ExplorViz
		PMD		
Durchschnitt	47,65	34,26	8,42	13,58
Unterschied		-28,06%		+61,28%
Minimum	23,04	29,43	3	4
Maximum	65,07	38,99	16	18
		Babsi		
Durchschnitt	31,55	29,14	9,40	13,04
Unterschied		-7,64%		+38,72%
Minimum	18,94	19,38	3	6
Maximum	43,20	41,56	18	18

Im ersten Experiment wurde mit ExplorViz eine um 61,28% bessere Punktzahl erreicht und im zweiten Experiment eine um 38,78% bessere Punktzahl. Wie schon für die benötigte Zeit wurden der Shapiro-Wilk-Test und der Levene-Test ausgeführt, um zu testen, ob die Bedingungen für den t-Test gegeben sind. Da dies der Fall ist, konnten für das erste Experiment ein Wert von 0,0015 und für das zweit ein Wert von 0,0007 berechnet werden, die beide kleiner sind als die gewählte Signifikanzgrenze. Daher kann die Nullhypothese für beide Experimente zurückgewiesen werden und stattdessen die alternative Hypothese $H2$ angenommen werden. Diese besagt, dass ein signifikanter Unterschied in der Korrektheit bei der Nutzung von Extravis und ExplorViz auftritt.

7.5. Bedrohungen der Validität

Dieser Abschnitt geht auf die verschiedenen Bedrohungen ein, die die Validität der Versuchsergebnisse gefährden könnten ([Wohlin u. a. 2012], [Shadish u. a. 2002]). Es wird weiterhin erläutert, was unternommen wurde, um die Bedrohungen zu unterbinden oder zumindest zu mindern.

7.5.1. Interne Validität

Die im Folgenden genannten Umstände bedrohen die interne Validität, da sie aus dem Experiment heraus entstehen.

Studienteilnehmer Da es sehr unwahrscheinlich ist, dass mehrere Menschen mit genau den gleichen Erfahrungen und dem gleichen Wissen existieren, ist es unmöglich, genau gleiche Studienteilnehmer zu finden. Um den Einfluss dieser Unterschiede auf die Ergeb-

nisse zu mindern, erfolgt die Zuteilung zu den untersuchten Visualisierungstools zufällig, mit der Annahme, dass sich so Gruppen mit ähnlichen Attributen bilden würden.

Den Probanden des ersten Experiments lag persönlich nichts daran, sich besonders große Mühe bei der Bearbeitung der Aufgaben zu geben. Um frühzeitigem Aufgeben oder halbherziger Arbeit entgegen zu wirken, wurde für die besten und schnellsten Probanden eine Belohnung ausgeschrieben. Dennoch kann dies nicht den Antrieb ersetzen, den ein Nutzer hat, der an dem visualisierten Programm wirklich interessiert ist. Die Probanden des zweiten Experiments hingegen hatten ein solches Interesse, da ihnen die Aufgabe bevorstand, Babsi zu überarbeiten und zu erweitern.

Mangelnde Kompetenz der Probanden wäre ein weiterer Grund für verfälschte Ergebnisse, da für das Verständnis eines Programms ein gewisser Grad an Programmiererfahrung notwendig ist. Dem steht jedoch entgegen, dass sich die Probanden, die PMD untersuchen sollten, selbst gute bis fortgeschrittene Programmiererfahrung zusprachen. Unter den Probanden, die Babsi untersucht haben, befanden sich hingegen auch Anfänger, doch da Babsi nur ein kleines Programm ist und von den selben Studenten in ihrem Kurs erwartet wird, Babsi zu überarbeiten, sollten auch sie nicht überfordert gewesen sein.

Instrumentierung Der für ExplorViz entwickelte Fragebogenmodus, der neben der Darstellung der Fragen und dem Speichern der Antworten auch für die Zeitmessung verantwortlich ist, konnte nach geringfügigen Änderungen auch für das Vergleichstool genutzt werden, sodass die Bestimmung der abhängigen Variablen auf gleiche Weise erfolgte.

Einführung in das Tool Dadurch, dass Tutorials genutzt wurden, hat jeder Versuchsteilnehmer, der das gleiche Tool bedient hat, die gleiche Anleitung erhalten. Die Nutzer von ExplorViz haben das im Verlauf dieser Arbeit implementierte interaktive Tutorial durchlaufen. Für die Extravis-Nutzer wurde ein bildliches und textuelles Tutorial zusammen mit einem Tutorialtrace zum Nachvollziehen der Schritte zur Verfügung gestellt. Dies war nötig, da keine ausgearbeitete Anleitung zu Extravis existiert. Es besteht daher möglicherweise eine Diskrepanz in der Qualität der Anleitung.

Durchschnittlich haben die Probanden 6:00 und 8:06 Minuten für ExplorViz beziehungsweise Extravis benötigt. Daran lässt sich erkennen, dass die Extravis-Probanden sich für das eigenverantwortliche Tutorial Zeit genommen haben, anstatt es nur zu überfliegen, obwohl es ihnen möglich gewesen wäre. Desweiteren kann ein Zeitverlust für die Bearbeitung der Aufgaben durch stark unterschiedliche Tutoriallängen durch diese Zeiten ausgeschlossen werden.

Wahrnehmung des Versuchsteilnehmers Um zu verhindern, dass die Versuchsteilnehmer sich durch Annahmen darüber, was für Ergebnisse erhofft sein könnten, beeinflussen lassen, wurde ihnen nicht mitgeteilt, welches Tool das untersuchte und welches das Vergleichstool war. Es wurden daher auch die gleichen vorbereitenden und abschließenden Fragen gestellt, wobei der Name des Tools an entsprechender Stelle ausgetauscht wurde.

Aufgaben Die Aufgaben wurden entsprechend der Aufgaben von Cornelissen [2009] erstellt. Daher kann ausgeschlossen werden, dass absichtlich Aufgaben gestellt wurden, die ExplorViz bevorzugen würden.

Eine weitere Bedrohung in Bezug auf die Aufgaben ist der Schwierigkeitsgrad. Werden zu schwere Aufgaben gestellt, kann dies den Unterschied in der Korrektheit bei Nutzung von Extravis oder ExplorViz beeinflussen. Dies war jedoch nicht der Fall, da von verschiedenen Probanden richtige Antworten abgegeben wurden. Des Weiteren wurden die Probanden im Anschluss an die Aufgaben aufgefordert, den Schwierigkeitsgrad jeder Aufgabe auf einer Skala von "zu einfach" bis "zu schwer" zu bewerten. Das Ergebnis dieser Bewertung war, dass die Aufgaben weder als zu schwer noch als zu leicht empfunden wurden.

Damit es bei der Bewertung der Antworten nicht zu einer Begünstigung der ExplorViz-Nutzer kommen kann, wurden die Antworten ohne Zuordnungsmöglichkeit zu den Teilnehmern in eine zufällige Reihenfolge gebracht und erst dann bewertet. Zusätzlich wurden die Antworten unabhängig von zwei Personen bewertet. Traten Diskrepanzen auf, wurde im Gespräch eine Einigung erzielt. Die Bewertung unterliegt daher nicht der Kontrolle einer einzigen Person. Dass bei unterschiedlichen Bewertungen maximal ein Punkt unterschied auftrat, deutet darauf hin, dass die Bewertung sachgemäß erfolgte.

Zeitvorgaben Den Probanden war bewusst, dass für die Durchführung des gesamten Experiments inklusive den Informationen zum visualiserten Programm und dem Tutorial ein Zeitraum von einer Stunde vorgesehen war. Weiterhin konnten sie unterhalb der jeweils aktuellen Frage sehen, wie viel Zeit für die Bearbeitung vorgesehen war und wurden auch darauf hingewiesen, wenn sie diesen Wert überschritten. Dies könnte dazu geführt haben, dass sich Studenten unter Druck gesetzt fühlten, eine unvollständige Antwort abzugeben oder die Bearbeitung der Aufgabe vollständig aufzugeben. Ob es bei einzelnen Probanden der Fall war, ist nicht auszuschließen, durchschnittlich wurde der empfundene Zeitdruck jedoch als angemessen für die Situation empfunden wurde. Außerdem konnte den Zeitmessungen entnommen werden, dass die vorgegebene Zeit mehrfach überschritten wurde, Probanden sich dadurch also nicht von der Beendigung einer Aufgabe haben abhalten lassen.

7.5.2. Externe Validität

Wenn die externe Validität bedroht ist, ist anzuzweifeln, ob die Ergebnisse der Studie sich verallgemeinern lassen. Im Folgenden werden mögliche Bedrohungen dargelegt und erläutert, wie ihnen entgegen gewirkt wurde.

Bezug zum visualisierten Programm Die Visualisierung eines Programms hängt stark von dem Programm ab, sowohl bezüglich der Größe als auch der Struktur des Programms. Aus diesem Grund wurde das zweite Experiment mit Babsi durchgeführt. Während PMD

ein mittelgroßes Programm mit guter Paketstruktur ist, handelt es sich bei Babsi um ein kleines Programm mit geringer Strukturierung. Die Ergebnisse hängen daher mit hoher Wahrscheinlichkeit nicht von der Größe des visualisierten Programms noch von der Art der Strukturierung ab.

Nutzungsdauer des Tools Da die Probanden sich nur etwa eine Stunde mit dem jeweiligen Tool beschäftigt haben, lässt sich keine Aussage darüber machen, ob bessere Vertrautheit andere Ergebnisse hervorbringen würden. Eine solche Vertrautheit würde sich bei mehrmaliger Nutzung einstellen und kann daher bei professioneller Anwendung angenommen werden. Um die Auswirkung der Nutzungsdauer zu untersuchen, sind deutlich längere Studien notwendig. Da die Probanden sich nur etwa eine Stunde mit dem jeweiligen Tool beschäftigt haben, lässt sich keine Aussage darüber machen, ob bessere Vertrautheit andere Ergebnisse hervorbringen würden. Eine solche Vertrautheit würde sich bei mehrmaliger Nutzung einstellen und kann daher bei professioneller Anwendung angenommen werden. Um die Auswirkung der Nutzungsdauer zu untersuchen, sind deutlich längere Studien notwendig.

Künstliche Situation Bei den Probanden handelt es sich um Studenten und nicht um professionelle, ausgebildete Entwickler. Es kann daher davon ausgegangen werden, dass sie nicht so viel Erfahrung haben, wie es die Nutzer bei professionellem Einsatz haben. Welchen Einfluss dieser Erfahrungsunterschied auf die Ergebnisse der Experimente hätte, lässt sich nur herausfinden, indem weitere Versuche mit erfahreneren Probanden durchgeführt werden.

Durch sechs Aufgaben kann zudem keine vollständige Abdeckung der möglichen Fragestellungen erreicht werden, denen sich ein Entwickler gegenüber sieht. Um dem entgegen zu wirken, wurde das Framework von Pacione u. a. [2004] verwendet und die dort festgelegten Aufgabenbereiche abgedeckt. Trotzdem kann von der Bearbeitung der speziellen Fragen nicht allgemein auf die Bearbeitung anderer Fragen geschlossen werden, sodass auch hier weitere Studien mit weiterreichenden Aufgaben sinnvoll sind, um die Ergebnisse zu verallgemeinern.

Kapitel 8

Evaluation

Im Anschluss an die Fragebögen der in Kapitel 7 beschriebenen Experimente erfolgten einige Fragen zu dem genutzten Programm, den Fragen und dem Tutorial. Dazu waren zwei Comboboxen gegeben, eine um zu bewerten, wie hilfreich das Tutorial war und eine weitere, um die Länge zu bewerten. Beide Comboboxen gehörten zu den Pflichtfeldern und mussten daher ausgefüllt werden. Zusätzlich wurde ein Textfeld für Kommentare zum Tutorial zur Verfügung gestellt, dessen Nutzung freiwillig war.

8.1. Empfundene Hilfe des Tutorials

Es gab fünf Antwortmöglichkeiten, um zu bewerten, wie hilfreich das Tutorial zum Bearbeiten der Aufgaben war. Den Antwortmöglichkeiten wurden die Werte 1 bis 5 zugewiesen, sodass ein genauer Durchschnittswert von 2,175 bestimmt werden konnte. Dies bedeutet, dass das Tutorial im allgemeinen als hilfreich empfunden wurde, mit einer geringen Tendenz zu einer ausreichenden Bewertung.

Die genaue Aufteilung der Antworten kann Tabelle 8.1 entnommen werden. Bei dieser fällt auf, dass kein Proband das Tutorial als gar nicht hilfreich empfunden hat. Allerdings haben 15% angegeben, nur wenig Hilfe durch das Tutorial erhalten zu haben.

Von diesen haben leider nur zwei einen Vorschlag gemacht, wie das Tutorial verbessert werden könnte, um diesen Umstand zu ändern. Den gegebenen Anmerkungen nach würde ein zurück-Button helfen, da ein Benutzer zu schnell die Anweisung ausgeführt hat, ohne den ganzen Text zu lesen, sodass er dies nicht mehr nachholen konnte, während der andere Benutzer absichtlich sehr schnell durch das Tutorial gegangen ist und erst hinterher bemerkt hat, dass ihm ein langsameres Vorgehen geholfen hätte.

Von den vier Probanden, die eine Bewertung als wenig hilfreich abgegeben haben aber keinen hilfreichen Kommentar hinterlassen haben, haben zwei Antworten abgegeben, die darauf hinweisen, dass sie nicht einmal versucht haben, die Aufgaben zu lösen. Die anderen beiden Probanden haben 11 bzw. 13 von 23 möglichen Punkten erreicht. Da die Durchschnittspunktzahl der Teilnehmer des Babsi-Experiments, die ExplorViz genutzt haben, bei 13,04 liegt, waren sie also trotz des als wenig hilfreich empfundenen Tutorials nur wenig schlechter als die anderen Probanden.

Tabelle 8.1. Bewertung der Hilfe durch 40 ExplorViz-Probanden

Bewertung	Anzahl
Sehr hilfreich	11
Hilfreich	17
Ausreichend	6
Wenig hilfreich	6
Nicht hilfreich	0

Tabelle 8.2. Bewertung der Länge durch 40 ExplorViz-Probanden

Bewertung	Anzahl
Zu lang	0
Lang	3
Angemessen	32
Kurz	4
zu kurz	1

8.2. Länge des Tutorials

Die Benutzer, die das Tutorial als lang empfunden haben, hätten knappere Texte bevorzugt und keine ausführliche Erklärung von Dingen, die für einige Personen offensichtlich sind. Dem entgegen stehen fünf Benutzer, die es als kurz oder zu kurz empfunden haben, sodass bei einer Überarbeitung sehr genau darauf geachtet werden muss, bei einer Kürzung von Texten keine Informationen einzubüßen und beim Hinzufügen von Schritten für weitere Funktionen möglichst wenige Funktionen doppelt ausführen zu lassen. Es lässt sich nicht vermeiden, dass manche Benutzer bestimmte Informationen für unnötig halten, weil sie selbstständig genug sind, während andere die Hilfe gerne annehmen, da sie nicht willkürlich Sachen ausprobieren. Ebenso sind manche Benutzer froh darüber, wenn ihnen der Sinn einer Funktion erklärt wird, während andere ihn von selbst erkennen und die Erklärung daher als unnötig erachten. Aufgrund dieser unterschiedlichen Ansprüche der Benutzer ist es auch nicht verwunderlich, dass das Tutorial durchschnittlich als angemessen, nämlich mit einem Wert von 3,075 beurteilt wurde. Die genaue Verteilung der gegebenen Bewertungen kann Tabelle 8.2 entnommen werden.

Weiterhin lässt sich bemerken, dass die drei Probanden, die das Tutorial für lang hielten, es trotzdem für ausreichend bis sehr hilfreich empfanden. Auf der anderen Seite fand nur der Proband, der es als zu kurz empfand, das Tutorial als wenig hilfreich, während die vier Probanden, die es kurz fanden, es ebenso als ausreichend bis sehr hilfreich bewertet haben.

8.3. Weitere Anmerkungen

Ein genanntes Problem, das durch die Möglichkeit, zu schon ausgeführten Schritten zurückzukehren gelöst werden könnte, trat bei der Aufforderung zum Hovern über einer Komponente auf. Zwei Probanden haben nach der Ausführung eines Schritts, auf den ein Hover-Schritt folgt, den Cursor zufällig über die Komponente bewegt, über der gehovert werden sollte. Dadurch haben sie unbeabsichtigt die Bedingung für den nächsten Schritt erfüllt, sodass dieser sofort ausgelöst wurde. Dadurch war es den Probanden nicht möglich, den erklärenden Text des entsprechenden Schritts zu lesen.

Die Problematik des Ausführens der geforderten Aktion bevor der Text zu ende gelesen wurde, wurde auch von Probanden angemerkt, die das Tutorial dennoch als ausreichend hilfreich befunden haben, was den Nutzen eines zurück-Buttons noch weiter hervorhebt, da es insgesamt fünf Probanden, also 12,5% zu erkennen gaben, dass sie einige Texte aus unterschiedlichen Gründen nicht zu ende lesen konnten.

8.4. Evaluation des Fragebogensystems

Bezüglich des Fragebogens kann angemerkt werden, dass ein Teilnehmer als Antwort auf eine Frage versucht hat, mithilfe einer SQL-Injection die zum Speichern der Antworten vermutete Datenbank zu löschen. Sollte daher vom Ablegen der Daten in CSV-Dateien zu einer Datenbank gewechselt werden, muss darauf geachtet werden, die Eingaben korrekt zu verarbeiten, damit SQL-Injections weiterhin nur als Antworttext und nicht als Teil der Anweisung behandelt werden.

Außerdem ist es aufgrund dessen, dass der Fragebogen in einem Browser dargestellt wird, mehrfach zu Problemen gekommen, wenn Benutzer versehentliche oder absichtliche die Zurück-Funktion des Browser ausgelöst haben. In dem Moment werden alle Informationen darüber, bei welcher Frage der Proband sich gerade befand, verworfen, sodass die vorherigen Fragen erneut beantwortet werden müssen. Da jede abgeschickte Antwort jedoch sofort an den Server gesendet und dort gespeichert wird, sind durch solche Probleme keine Daten verloren gegangen. Dadurch war es nicht nötig, dass die Antworten beim Blättern zu der aktuell bearbeiteten Aufgabe sinnvoll waren, sondern nur benötigt wurden, da ein Antwortfeld nicht leer abgesendet werden kann. Um solche Probleme weitestgehend zu vermeiden, muss das default-Verhalten der Backspace-Taste unterbunden werden, da diese ein zurückblättern in der Browser-History auslöst. Die direkte Nutzung des Zurück-Buttons im Browser lässt sich allerdings nicht verhindern, sodass über Möglichkeiten nachgedacht werden müsste, wie die Nutzerdaten auch beim Verlassen über das Verlassen der Seite hinaus gespeichert werden können.

8.5. Vergleich zu Extravis-Tutorial

Bei dem Extravis-Tutorial, das speziell für die Experimente angelegt wurde, da Extravis keine Anleitung beliegt, handelt es sich um ein textuelles, nicht interaktives Tutorial. Stattdessen wurde den Benutzern ein Beispieltrace zur Verfügung gestellt, an dem sie die beschriebenen Funktionen testen konnten. Da sie aber im Gegensatz zum ExplorViz-Tutorial selbst entscheiden konnten, ob sie den Anweisungen folgen oder nur die Texte lesen, unterscheiden sich die beiden Tutorials dadurch, wie viel Kontrolle sie dem Benutzer über den Vorgang geben.

Die Länge wurde ähnlich bewertet wie die des ExplorViz-Tutorials, mit einem Durchschnittswert von 3,15. Die empfangene Hilfe wurde jedoch nur noch mit 2,9 bewertet, was nur noch ausreichender Hilfe entspricht. Es lässt sich jedoch nicht sagen, ob dieser Unterschied seine Ursache in den gegebenen Erklärungen findet, oder ob er durch die mangelnde Kontrolle darüber, was der Benutzer während des Tutorials tut, zustande kommt.

Kapitel 9

Verwandte Arbeiten

In diesem Kapitel werden einige verwandte Arbeiten vorgestellt und anschließend aufgezeigt, wodurch sich die beschriebenen Experimente von diesen Arbeiten unterscheiden.

Eine direkte Verbindung kann zu Cornelissen u. a. [2011] gezogen werden, da das Trace Visualisierungstool Extravis im Vergleich zur Sourcecode-Analyse mit Eclipse verglichen wurde. Wie schon in Kapitel 8 erwähnt, wurde aus diesem Grund Wert darauf gelegt, den Vergleich zwischen ExplorViz und Extravis möglichst analog zu dieser Studie durchzuführen. Es wurde ein Programm (Checkstyle) von zwei verschiedenen Gruppen untersucht, wobei die eine Gruppe nur Eclipse nutzen konnte, während der anderen Eclipse und Extravis zur Verfügung standen. Für beide Gruppen wurde gemessen, wie viel Zeit sie zur Bearbeitung der Aufgaben brauchten, und wie gut die gegebenen Antworten waren.

Die Gruppe, die zusätzlich Extravis genutzt hat, bearbeitet die Aufgaben in durchschnittlich 20,51% weniger Zeit und Cornelissen u. a. [2011] zeigte Gründe auf, deren Beseitigung die Beschleunigung noch weiter steigen könnten. Bei der Korrektheit wurde sogar ein Unterschied von 43,14% zu Gunsten von Extravis festgestellt, sodass von einer deutlichen Verbesserung gesprochen werden kann.

Quante [2008] hat in einem kontrollierten Experiment untersucht, ob "Dynamic Object Process Graphs"(DOPG) einen positiven Einfluss auf das Programmverständnis haben. Da nur wenige Versuchsteilnehmer zur Verfügung standen, wurden zwei verschiedene Programme (ArgoUML und Gantt) untersucht, wobei jeder Teilnehmer in der Kontrollgruppe für das eine und in der Versuchsgruppe für das andere Programm war, um Lerneffekte als Ursache für die Ergebnisse auszuschließen. Dadurch entstanden zwei verschachtelte Experimente.

Der Kontrollgruppe stand zur Bearbeitung der Aufgaben Eclipse zur Verfügung, während die Experimentgruppe Eclipse mit einem DOPG Eclipse Plugin nutzen konnte.

Es wurde herausgefunden, dass abhängig vom untersuchten System die Bearbeitungszeit mit DOPGs kürzer sein kann, aber nicht kürzer sein muss. Die Veränderung der Korrektheit war ebenso abhängig vom untersuchten System. Der Nutzen von DOPGs steht daher in direktem Zusammenhang mit dem Programm und den Informationen, die der Nutzer gewinnen will.

Bei der Durchführung dieses Experiments wurde Exclipse (*Exclipse*) zur Aufzeichnung der Daten genutzt. Exclipse ist ein Plugin für Eclipse, das die Durchführung von Experi-

menten erleichtern soll. Zu diesem Zweck können Fragen und Aufgaben sowie Zeitangaben zur maximalen Bearbeitungsdauer direkt in Eclipse integriert werden. Die gegebenen Antworten sowieso Logging-Daten werden gespeichert, sodass der Experimentleiter sie später auswerten kann. Zur Konfiguration wird XML genutzt [*Exclipse*]. Sofern Eclipse sowieso Bestandteil der zu nutzenden Tools sowohl der Kontroll- als auch der Experimentgruppe ist, stellt Exclipse daher eine gute Möglichkeit dar, um ohne großen Aufwand einen automatischen Fragebogen zu erstellen. Soll Eclipse hingegen nur von einer Gruppe genutzt werden, bietet es sich nicht an, da die Nutzung von Eclipse nur zur Verwendung eines Fragebogens die Performance des Rechners negativ beeinträchtigen kann.

In einer Studie von Wettel u. a. [2011] wurden kontrollierte Experimente durchgeführt, um herauszufinden, ob die Nutzung des Tools CodeCity, einem visuellen Tool zur Darstellung eines Software Systes als dreidimensionale Stadt, zu besserem Programmverständnis führt als die Nutzung von Eclipse und Excel-Tabellen, die über die benötigten Informationen verfügen. Dazu wurden ein mittelgroßes Programm (FindBugs) und ein großes Programm (Azureus) untersucht, sodass Unterschiede des Nutzens bei verschiedenen Programmgrößen aufgedeckt werden können.

Die Studie erbrachte das Ergebnis, dass die Korrektheit der Antworten bei Nutzung von CodeCity 24,26% größer war, als bei der Nutzung von Eclipse und Excel. Unabhängig davon, welches Tool zur Analyse genutzt wurde, wurde außerdem festgestellt, dass bessere Ergebnisse für das mittelgroße Programm erzielt wurden. Weiterhin war die Bearbeitungszeit der Benutzer von CodeCity um durchschnittlich 12,01% kürzer, sodass auch dort eine Verbesserung festgestellt werden konnte. Ebenso wie bei der Korrektheit der Antworten war die Bearbeitungszeit für das mittelgroße Programm vom genutzten Tool unabhängig deutlich geringer als für das große.

Storey u. a. [1997] haben untersucht, welchen Einfluss drei verschiedene Tools zur Gewinnung von Programmverständnis (Rigi, SHriMP, SNiFF+) auf das Programmverständnis der Studenteilnehmer hatten. Dabei wurde besonderer Wert darauf gelegt, dass solche Tools dem Programmierer keine Strategie aufzwingen sollen, sondern seine bevorzugte Strategie unterstützen sollen. Ob dies der Fall ist oder ob Strategien behindert werden, war Mittelpunkt der Untersuchungen.

Jedem Versuchsteilnehmer wurde ein Tool zugewiesen, mit denen das gleiche Programm (ein Monopoly Spiel) untersucht wurde. Über alle Aufgaben hinweg konnte beobachtet werden, dass die Tools bei der Bearbeitung der Aufgaben geholfen haben, also Features zur Verfügung stellen, die der bevorzugten Strategie des Nutzers entgegenkamen. Trotzdem kam es bei jedem der drei Tools zu Problemen mit einzelnen Strategien für bestimmte Aufgaben, die entweder nicht oder nur sehr schwer umgesetzt werden konnten. Dies lag zum Teil an der Art, wie das Tool das untersuchte Programm darstellt, und zum Teil an fehlenden Features.

In der Studie von Marcus u. a. [2005] wird die Bedienbarkeit von sourceviewer3D (sv3D) untersucht, sowohl um die Auswirkungen der Nutzung zu bestimmen, als auch um Feedback zu sammeln, anhand dessen sv3D verbessert werden soll.

Eine Hälfte der Probanden beantwortete Fragen in Bezug auf das ausgewählte Testprogramm mit Hilfe von sv3D, während die andere Hälfte nur den Sourcecode und tabellarische Informationen (Durchschnittswerte, Verschachtelungstiefen, u.ä.) zu jeder Datei und jede Zeile zur Verfügung hatte.

Es stellte sich heraus, dass kein signifikanter Unterschied in der Korrektheit der Antworten durch die Nutzung von sv3D erreicht werden konnte und sogar mehr Zeit für die Beantwortung benötigt wurde- Daraus lässt sich erkennen, dass Benutzer Schwierigkeiten haben, wenn sie neue Technologien einsetzen sollen, selbst wenn sie selbst nicht das Gefühl hatten, Probleme bei der Benutzung des noch unbekannten Programm zu haben.

Sharif u. a. [2013] haben die Auswikungen des Tools SeeIT 3D als Zusatz zur Nutzung von Eclipse zum Bearbeiten typischer Aufgaben untersucht. Daten wurden dabei einerseits über einen Fragebogen gesammelt, andererseits wurde aber auch ein Eye-Tracker genutzt, um das Verhalten der Probanden analysieren zu können. Den Probanden der Experimentgruppe stand zur Bearbeitung der Aufgaben Eclipse mit dem SeeIT 3D Plugin zur Verfügung, während die Kontrollgruppe kein Visualisierungstool sondern nur Eclipse genutzt hat. Gemessen wurden wie üblich die Bearbeitungszeit und die Genauigkeit der Antworten, zusätzlich jedoch für einen Teil der Probanden auch das Verhalten mithilfe des Eye-Trackers. Untersucht werden sollte, ob SeeIT 3D einem Entwickler bei verschiedenen Aufgaben hilft, und ob Unterschiede zwischen Experten und Anfängern auftreten. Die Aufgaben wurden in die Kategorien *einen Überblick verschaffen*, *neue Features* und *Fehlerbehebung* unterteilt. Der Nutzen von SeeIT 3D wurde in Zeit, Genauigkeit der Antworten und benötigter visueller Anstrengung gemessen. Als zu untersuchendes System wurde GanttProject gewählt. Bei den Probanden handelte es sich um Studenten, die nach Beantworten eines Fragebogens bezüglich ihrer Fähigkeiten in Experten und Anfänger unterteilt wurden, und innerhalb dieser Gruppen zufällig in Experiment- und Kontrollgruppe aufgeteilt wurden, sodass vier Gruppen entstanden. Nur bei den Aufgaben zur Übersichtsverschaffung trat höhere Genauigkeit in den Experimentgruppen auf, die Genauigkeit der Fehlerbehebungsaufgaben und Aufgaben zur Implementierung neuer Features unterschied sich nicht. Das selbe Ergebnis trat für die benötigte Zeit zur Bearbeitung der Aufgaben auf. Die visuelle Anstrengung wurde nur anhand von 18 Probanden gemessen und es traten keine Unterschiede zwischen der Kontrollgruppe und der Experimentgruppe auf. Abgesehen von der zu erwartenden höheren Genauigkeit der Expertengruppe gegenüber der Anfängergruppe wurde keine signifikante Auswirkung des Erfahrungsgrades auf die zusätzliche Nutzung von SeeIT 3D gegenüber der Nutzung von Eclipse festgestellt.

Im Gegensatz zu den Studien, die in den vorgestellten Arbeiten durchgeführt wurden, wurden in der in dieser Arbeit beschriebenen Studien zwei Trace Viusalisierungstool

miteinander verglichen. Die genannten Studien haben hingegen entweder ein Visualisierungstool mit der Arbeit mit einer IDE verglichen, also herausfinden wollen, ob die Visualisierung einen Vorteil bringt, oder Tools verglichen, die durch verschiedene Ansätze Programmverständnis vermitteln wollen.

Da aus den bekannten Studien hervorgegangen ist, dass Visualisierungen grundsätzlich hilfreich sind, wurde auf einen erneute Studie dieser Art verzichtet und stattdessen ein Tool zum Vergleich ausgewählt, das den gleiche Ansatz für die Vermittlung des Programmverständnis nutzt, nämlich die Darstellung eines Traces. Die Tools unterscheiden sich also nur durch die gewählte Visualisierungsart sowie die Bedienung und einzelne Features. Um bessere Rückschlüsse auf die Visualisierung ziehen zu können, wurden einige erweiterte Features von ExplorViz für die Dauer der Experimente deaktiviert, damit diese keinen Grund für auftretende Unterschiede bieten.

Mit Ausnahme von Quante [2008] und Sharif u. a. [2013] wurde in keiner der Studien erwähnt, wie die Zeitmessung vorgenommen wurde, sodass Fragen zur Herkunft und Genauigkeit der angegebenen Zeiten aufkommen können. Während Quante [2008] ein zusätzliches Plugin genutzt hat, ist der Fragebogen inklusive Zeitmessung im Zuge dieser Arbeit in ExplorViz integriert worden, sodass keine zusätzliche Software benötigt wird.

Auch wurden verschiedene, ausführlichere Arten der Einweisung in die zu nutzenden Tools gegeben. Genutzte Einführungsmöglichkeiten, bei denen gewährleistet wurde, dass alle Benutzer die selben Informationen bekommen, waren längere Präsentationen, kleine Übungsaufgaben anhand von kleineren Datenmengen und in einem Fall sogar zusätzliche Online-Übungen, die vor der eigentlichen Übungsstunde bearbeitet werden sollen. Weniger Kontrolle darüber, dass keine Beeinflussung der Benutzer stattfindet, gab es in den Fällen, in denen Benutzer beim Bearbeiten von Übungsaufgaben konkrete Hilfe von einem Experimentleiter erhalten haben, da es dem Leiter dabei möglich wäre, den Probanden der Experimentgruppe unbewusst mehr Hilfe zu geben als denen der Kontrollgruppe.

Sowohl das Tutorial als auch der Fragebogen wurden speziell auf ExplorViz abgestimmt - wobei sich herausgestellt hat, dass die Kopplung des Fragebogens leicht von ExplorViz gelöst werden kann. Daher kann mit Sicherheit gesagt werden, dass keine Replikation einer anderen Arbeit vorliegt, auch wenn sicherlich schon viele Fragebögen (z.B. [*Exclipse*]) und Tutorials entworfen und implementiert wurden.

Kapitel 10

Fazit und Ausblick

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst. Außerdem sind während der Experimente einzelne Stellen aufgefallen, an denen das Tutorial und der Fragebogen noch verbessert werden können. Diese werden ebenfalls in diesem Kapitel angesprochen und es werden Vorschläge gemacht, wie sie umgesetzt werden könnten.

10.1. Fazit

In dieser Arbeit wurden verschiedene Ansätze zur Umsetzung einer automatischen Anleitung einer Versuchsperson während eines kontrollierten Experiments in ExplorViz durch ein interaktives Tutorial sowie einen elektronischen, in das Programm integrierten Fragebogen diskutiert. Aus den vorgestellten Möglichkeiten wurden die jeweils geeignetsten ausgewählt und daraus ein Konzept zusammengestellt, das anschließend in der Implementierung umgesetzt wurde. Da die Einweisung in ExplorViz automatisch erfolgt, werden Bedrohungen der Validität durch Beeinflussung der Probanden durch Versuchsleiter gemindert. Der elektronische Fragebogen unterstützt die Durchführung eines Experiments durch automatische Zeitmessung und Speicherung der gegebenen Antworten in einem Format, das die Auswertung erleichtern soll.

Beide Funktionalitäten sind in zwei kontrollierten Experimenten zum Einsatz gekommen, deren Aufbau, Ablauf und Ergebnisse zum besseren Verständnis ebenfalls erläutert wurden. Aufgrund der Meinungsumfrage im Anschluss an die Experimente lässt sich feststellen, dass das Tutorial gut aufgenommen und als hilfreich empfunden wurde. Auch der Fragebogen hat sich im Zuge der Experimente als nützlich erwiesen und das gewählte Format zum Speichern der Daten konnte mit Hilfe von Skripten problemlos in verschiedene, zur Auswertung benötigte Formate konvertiert werden.

10.2. Gewonnene Erkenntnisse

Bei der Umsetzung des Tutorialkonzepts hat sich die Wahl für ein modulares System als gute Entscheidung herausgestellt. Während es bei einem Programm, dessen Entwicklung definitiv abgeschlossen ist, unnötig ist, erspart ein modulares System bei einem Programm in Entwicklung viel Arbeit, wenn neue Features hinzugefügt werden oder die Logik hinter einem Feature geändert wird. Dadurch war es nicht notwendig, das gesamte Tutorial zu

überarbeiten. Stattdessen war es ausreichend, einzelne Bestandteile anzupassen oder neue Schritte anzulegen und diese in den Ablauf einzufügen.

Dass ExplorViz als Webanwendung implementiert wurde und daher im Browser läuft, brachte sowohl für das Tutorial als auch den Fragebogen einige Herausforderungen während der Experimente mit sich. Es ist zwar hilfreich, dass Tastatureingaben ohne viel Aufwand erkannt und verarbeitet werden können, allerdings gibt es einige Tasten, die bestimmte Browserfunktionen aktivieren, die den Ablauf unterbrechen. Drückt der Nutzer Backspace, während er sich nicht in einem Eingabefeld befindet, wird im Browser die *eine Seite zurückgehen*-Funktion ausgelöst und damit die ExplorViz-Seite verlassen. Dabei gehen alle clientseitig gespeicherten Nutzerinformationen verloren, wozu der Fortschritt im Tutorial bzw. Fragebogen gehört. Als Folge dessen muss der Benutzer das Tutorial erneut beginnen bzw. bekommt die schon bearbeiteten Aufgaben erneut angezeigt.

Bei derartigen Problemen war die Entscheidung hilfreich, Userdaten direkt nach der Eingabe an der Server zu senden, um den Verlust von Daten zu vermeiden. Wären die gegebenen Antworten clientseitig gesammelt und erst nach Beendigung des Fragebogens an den Server geschickt worden, wären die Antworten genauso wie die Fortschrittsdaten beim Verlassen der Seite verloren gegangen, sodass der Nutze die Fragen erneut hätte beantworten müssen. Dies war nicht der Fall, sodass die Fragen nach Verlassen der Seite zwar erneut angezeigt werden, aber nicht korrekt beantwortet werden müssen.

Die lose Kopplung des Fragebogens mit der Funktionalität von ExplorViz, die nur durch die Möglichkeit zur Angabe von Zeiten besteht, bis zu denen die Aufzeichnung abgespielt werden soll, hat sich ebenfalls ausgezahlt, da es dadurch möglich war, den Fragebogen mit nur geringen Anpassungen auch für das Vergleichstool zu benutzen.

10.3. Ausblick

Um die Herausforderung des Fortschrittsdatenverlusts bei unabsichtlichem Verlassen der Seite zu bewältigen, wäre es denkbar, Cookies im Browser zu hinterlegen, die Informationen darüber beinhalten, welche Frage der Nutzer gerade bearbeitet. Diese müssten beim Fortschreiten ebenso aktualisiert werden wie der interne Zähler und bei Beendingung des Fragebogens gelöscht werden, sodass der aufzurufende Zustand anhand der Cookies immer eindeutig bestimmt ist. Beim Aufrufen des Fragebogens würde dann zuerst nachgefragt werden, ob ein entsprechendes Cookie vorhanden ist und, falls dies der Fall ist, an die entsprechende Stelle des Fragebogens gesprungen werden. Ist kein Cookie vorhanden, würde der Fragebogen am Anfang beginnen.

Auch mit einer solchen Sicherung des Fortschritts wäre es wünschenswert, wenn die Seite nicht versehentlich durch das Drücken einer Taste verlassen werden würde. Dazu muss das Standardverhalten solcher Tasten (zum Beispiel Backspace) unterbunden werden. Dabei ist allerdings zu beachten, dass sich manche Tasten abhängig von der Umgebung unterschiedlich verhalten und daher nicht jede Umgebung betroffen ist. Zum Beispiel sollte Backspace innerhalb eines Eingabefeldes weiterhin zum Löschen der Eingabe genutzt

werden können.

Zum Analysieren des Nutzerverhaltens während der Bearbeitung der Aufgaben würde es sich anbieten, den User Trace, der von ExplorViz erzeugt wird, mit den Antworten zu verbinden. Eine Möglichkeit dazu wäre ein Menüpunkt, der automatisch für jede Aufgabe eine Logging-Datei erzeugt, die alle Logging-Daten aus dem Zeitraum enthält, in dem die Aufgabe bearbeitet wurde.

Wenn weitere Funktionalitäten zu ExplorViz hinzugefügt werden, müssen diese in das Tutorial eingepflegt werden. Außerdem ist zwar die Möglichkeit zur Unterstützung mehrerer Sprachen vorhanden, indem eine Auswahloption besteht, aufgrund derer die einzulesenden Dateien bestimmt werden, doch liegen nur englische Texte vor. Sollen weitere Sprachen ausgewählt werden können, müssen die Texte erst übersetzt und in das Dateisystem eingepflegt werden.

Von den Probanden des Experiments wurde ein Zurück-Button für das Tutorial erwünscht. Wenn ein solcher hinzugefügt werden sollte, muss darauf geachtet werden, dass es möglich sein muss, die Bedingung des vorherigen Schritts erneut auszuführen, da es Ausführungsreihenfolgen möglich sein können, in denen dies nicht ohne weiteres möglich ist, wenn Pakete geschlossen oder die Ansicht gewechselt wurde. Eine Möglichkeit, diese Problematik zu umgehen, wäre, beim erneuten Anzeigen eines Schrittes nur den Text anzuzeigen und auch für solche Schritte, die normalerweise eine bestimmte Aktion benötigen, einen Weiter-Button anzuzeigen. Dazu müsste eine Fallunterscheidung zwischen dem ersten Anzeigen des Textes und dem erneuten Anzeigen getroffen werden, die gegebenenfalls den zusätzlichen Button anzeigt und die Aktion zum Fortfahren aktiviert oder deaktiviert.

Literaturverzeichnis

[*Bootstrap*] Bootstrap. Letzter Zugriff 27.07.2014. URL: http://getbootstrap.com/. (Siehe Seite 10)

[Charney und Reder 1986] D. H. Charney und L. M. Reder. Designing Interactive Tutorials for Computer Users. *Human-Computer Interaction* 2 (1986), Seiten 297–317. (Siehe Seiten 13, 14)

[Charney u. a. 1988] D. H. Charney, L. M. Reder und G. W. Wells. Studies of Elaboration in Instructional Texts. *Effective Documentation: What We Have Learned From Research* (1988), Seiten 47–72. (Siehe Seite 14)

[Cornelissen u. a. 2011] B. Cornelissen, A. Zaidman und A. van Deursen. A Controlled Experiment for Program Comprehension Through Trace Visualization. *IEEE Transactions on Software Engineering* 37.3 (2011), Seiten 341–355. (Siehe Seite 67)

[Cornelissen 2009] S. G. M. Cornelissen. Evaluating Dynamic Analysis Techniques for Program Comprehension. Dissertation. Delft University of Technology, 2009. (Siehe Seiten 49–51, 56 und 60)

[Efftinge u. a. 2012] S. Efftinge, M. Eysholdt, J. Köhnlein, S. Zarnekow, R. Massow, W. Hasselbring und M. Hanus. Xbase: Implementing Domain-Specific Languages for Java. In: *Proceedings of the 11th International Conference on Generative Programming and Component Engineering (GPCE 2012)*. ACM, 2012, Seiten 112–121. (Siehe Seite 6)

[*Exclipse*] Exclipse. Letzter Zugriff 19.08.2014. URL: http://exclipse.sourceforge.net/. (Siehe Seiten 67, 68 und 70)

[Fittkau u. a. 2013] F. Fittkau, J. Waller, C. Wulf und W. Hasselbring. Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach. In: *Proceedings of the 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*. 2013. (Siehe Seiten 1 und 5)

[Fittkau u. a. 2015] F. Fittkau, S. Finke, W. Hasselbring und J. Waller. Comparing Trace Visualizations for Program Comprehension through Controlled Experiments. In: *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*. submitted. ICSE. 2015. (Siehe Seite 49)

[*GWT*] GWT. Letzter Zugriff 26.03.2014. URL: http://www.gwtproject.org/overview.html. (Siehe Seite 8)

[*GWT RPC-Tutorial*] GWT RPC-Tutorial. Letzter Zugriff 26.03.2014. URL: http://www.gwtproject.org/doc/latest/tutorial/RPC.html. (Siehe Seite 8)

[*JQuery*] JQuery. Letzter Zugriff 27.07.2014. URL: http://jquery.com/. (Siehe Seite 9)

[*JQuery UI*] JQuery UI. Letzter Zugriff 27.07.2014. URL: http://jqueryui.com/. (Siehe Seite 9)

[Kosche 2013] M. Kosche. Tracking User Actions for the Web-Based Front End of ExplorViz. Bachelorarbeit. Kiel University, 2013. (Siehe Seite 3)

[Levene 1960] H. Levene. Robust Tests for Equality of Variances. *Contributions to probability and statistics: Essays in honor of Harold Hotelling* 2 (1960), Seiten 278–292. (Siehe Seite 57)

[Marcus u. a. 2005] A. Marcus, D. Comorski und A. Sergeyev. Supporting the Evolution of a Software Visualization Tool Through Usability Studies. In: *Proceedings of the 13th International Workshop on Program Comprehension (IWPC 2005)*. IEEE. 2005, Seiten 307–316. (Siehe Seite 69)

[Nixon u. a. 2009] J. M. Nixon, M. Slebodnik und C. F. Riehle. Creating Online Tutorials at Your Libraries: Software Choices and Practical Implications. *Reference & User Services Quarterly* 49 (2009), Seiten 33–37. (Siehe Seiten 14, 15)

[Pacione u. a. 2004] M. J. Pacione, M. Roper und M. Wood. A Novel Software Visualisation Model to Support Software Comprehension. In: *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004)*. 2004, Seiten 70–79. (Siehe Seiten 50, 51 und 61)

[Porst 2008] R. Porst. Fragebogen, Ein Arbeitsbuch. Springer VS, 2008. (Siehe Seite 17)

[*PubFlow*] PubFlow. Letzter Zugriff 12.09.2014. URL: `http://www.pubflow.de`. (Siehe Seite 24)

[Quante 2008] J. Quante. Do Dynamic Object Process Graphs Support Program Understanding? - A Controlled Experiment. In: *Proceedings of the 16th IEEE International Conference on Program Comprehension (ICPC 2008)*. 2008, Seiten 73–82. (Siehe Seiten 67 und 70)

[Shadish u. a. 2002] W. R. Shadish, T. D. Cook und D. T. Campbell. Experimental and Quasi-Experimental Designs for Generalized Causal Inference. Wadsworth Cengage learning, 2002. (Siehe Seiten 10, 11 und 58)

[Shapiro und Wilk 1965] S. S. Shapiro und M. B. Wilk. An Analysis of Variance Test for Normality (complete samples). *Biometrika* (1965), Seiten 591–611. (Siehe Seite 57)

[Sharif u. a. 2013] B. Sharif, G. Jetty, J. Aponte und E. Parra. An Empirical Study Assessing the Effect of SeeIT 3D on Comprehension. In: *Proceedings of the 1st IEEE Working Conference on Software Visualization (VISSOFT 2013)*. IEEE. 2013, Seiten 1–10. (Siehe Seiten 69, 70)

[Storey u. a. 1997] M.-A. Storey, K. Wong und H. A. Muller. How Do Program Understanding Tools Affect How Programmers Understand Programs? In: *Proceedings of the 4th Working Conferenceon Reverse Engineering (WCRE 1997)*. IEEE. 1997, Seiten 12–21. (Siehe Seite 68)

[*.validate*] .validate. Letzter Zugriff 17.07.2014. URL: `http://jqueryvalidation.org`. (Siehe Seiten 40, 41 und 43)

[Wettel und Lanza 2007] R. Wettel und M. Lanza. Visualizing Software Systems as Cities. In: *Proceedings of the 4th International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007)*. IEEE. 2007, Seiten 92–99. (Siehe Seite 5)

[Wettel u. a. 2011] R. Wettel, M. Lanza und R. Robbes. Software Systems as Cities: A Controlled Experiment. In: *Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011)*. ACM. 2011, Seiten 551–560. (Siehe Seite 68)

[Wohlin u. a. 2012] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell und A. Wesslén. Experimentation in Software Engineering. Springer, 2012. (Siehe Seite 58)

[*Rigi user's manual - version 5.4.4*] K Wong. Rigi user's manual - version 5.4.4. Letzter Zugriff 13.09.2014. URL: http://www.rigi.cs.uvic.ca/downloads/pdf/rigi-5_4_4-manual.pdf. (Siehe Seite 54)

[*Xtend User Guide*]. Xtend User Guide. URL: https://www.eclipse.org/xtend/documentation/2.3.0/Documentation.pdf. (Siehe Seite 6)

Anhang

Dieses Kapitel gibt eine Übersicht über die Daten, die auf der beiliegenden DVD vorhanden sind.

Daten auf der DVD

Auf der DVD enthalten sind drei Ordner mit folgenden Inhalten:

Bilder Enthält alle in der Masterarbeit verwendeten Grafiken.

PDF Enthält die vorliegende Masterarbeit im PDF-Format.

ExplorViz Enhält den ExplorViz Sourcecode.