Visualisierung holistischer Code-Metriken von Code Cities in SEE

Bachelorarbeit

Maximilian Wick

Matrikelnummer: 4566504

29. Dezember 2022

Fachbereich 3 — Mathematik und Informatik
Studiengang Informatik

1. Gutachter: Prof. Dr. Rainer Koschke
2. Gutachter: Dr. Hui Shi

ZUSAMMENFASSUNG

In dieser Arbeit geht es darum, wie man Code-Metriken, die einen Quellcode in seiner Gesamtheit betreffen, darstellen kann. Die Darstellung ist in rein textueller Form möglich, vermutlich jedoch nicht optimal. Hier sollte untersucht werden, wie man die Darstellung besser gestalten kann. Die Implementierung geschah als Erweiterung des Projektes *SEE*, welches zur Software-Visualisierung dient.

SEE: *Das steht für* ***S****oftware* ***E****ngineering* ***E****xperience und ist ein Programm, mit dem Software in Form von* Code Cities *visualisiert wird. Dabei baut SEE auf der Unity-Engine auf.*

ERKLÄRUNG

Ich versichere, diese Arbeit — sofern dies nicht explizit anders gekennzeichnet wurde — ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, den 29. Dezember 2022

Maximilian Wick

INHALTSVERZEICHNIS

1 EINFÜHRUNG

In der AG Softwaretechnik an der Universität Bremen gibt es ein Projekt namens SEE. Das ist ein Programm, mit dem man Software in Form von *Code Cities* visualisieren kann. Die Visualisierung als Code City bedeutet, dass jedes Element des Quellcodes der Software einzeln dargestellt wird. Auch die Eigenschaften der Elemente werden für jedes Element einzeln angezeigt. Dazu wird beispielsweise die Größe (in der Anzahl an Zeilen) der Elemente durch die Größe der Häuser der Code City dargestellt.

Code City: *In der Code-City-Metapher werden Softwarekomponenten durch Gebäude in einer Stadt repräsentiert, wobei die Eigenschaften dieser Gebäude verschiedene Metriken der Software ausdrücken können — z. B. könnte die Höhe eines Gebäudes der Anzahl der Codezeilen entsprechen.*

1.1 ZIEL DER ARBEIT

Bisher wurden in SEE hauptsächlich Metriken visualisiert, die die Eigenschaften jedes Elements einzeln bemessen, wie auch im vorigen Absatz beschrieben. Ziel dieser Arbeit war, SEE um eine Komponente zu erweitern, in der Metriken zu den Eigenschaften einer gesamten Code City visualisiert werden (solche Metriken werden in dieser Arbeit als „holistisch“ bezeichnet). Eine solche Eigenschaft könnte beispielsweise die Anzahl an Elementen einer Code City sein. Folgende Frage sollte in dieser Arbeit beantwortet werden:

> Wie lassen sich Metriken zu Eigenschaften ganzer Code Cities geeignet in SEE visualisieren?

Allerdings ist der Begriff „geeignet“ sicherlich zu allgemein. Deswegen wird die Fragestellung nun in drei Unterpunkte aufgeteilt.

- Die Darstellung soll einen Mehrwert bieten.
- Die Darstellung soll sich gut bedienen lassen.
- Die Darstellung soll ansprechend sein.

1.2 ÜBERBLICK ÜBER DEN INHALT DER ARBEIT

GRUNDLAGEN, VERWANDTE ARBEITEN In Kapitel 2 werden Grundlagen erläutert, die man kennen sollte, um diese Arbeit verstehen zu können. Außerdem wird kurz der Stand der Wissenschaft zum Thema dieser Arbeit betrachtet.

KONZEPT In Kapitel 3 wird abstrakt wiedergegeben, welche Faktoren die Implementierung beeinflusst haben, die die Fragestellung aus Abschnitt 1.1 beantworten sollte.

IMPLEMENTIERUNG In Kapitel 4 wird etwas anschaulicher und auch technischer beschrieben, wie das Konzept konkret implementiert wurde.

NUTZERSTUDIE Um zu zeigen, ob das *Framework*, das implementiert wurde, die Forschungsfrage aus Abschnitt 1.1 erfolgreich beantwortet, wurde eine Studie durchgeführt. Wie diese Studie aufgebaut wurde und was herausgekommen ist, wird in Kapitel 5 dargestellt.

Framework: *Das Framework, das im Rahmen dieser Bachelorarbeit implementiert wurde, um Werte holistischer Metriken zu visualisieren.*

ZUSAMMENFASSUNG, AUSBLICK Am Ende werden in Kapitel 6 nochmal die Ergebnisse dieser Arbeit insgesamt betrachtet. Außerdem wird dort das Vorgehen bei der Durchführung dieser Bachelorarbeit reflektiert.

2

GRUNDLAGEN UND VERWANDTE ARBEITEN

In diesem Kapitel wird grundlegendes Wissen vermittelt, welches nicht unbedingt alle Leser haben. Außerdem betrachten wir kurz den Stand der Wissenschaft zu diesem Thema.

2.1 GRUNDLAGEN

MODELLIERUNG Die hierarchische Struktur eines Code-Projekts unterscheidet sich von Programmiersprache zur Programmiersprache, verschiedene Programmiersprachen verwenden verschiedene Elemente zur Strukturierung. Das können Namespaces, Pakete, Module, Klassen, Methoden und so weiter sein. Da wir uns nicht auf eine Programmiersprache beschränken, wollen wir davon abstrahieren. Wir modellieren daher den Quellcode als mathematischen Graph, genauer gesagt als Baum. Der Wurzelknoten könnte dann zum Beispiel das gesamte Projekt sein (oder der Ordner, der dieses enthält), die inneren Knoten könnten Namespaces oder Klassen sein und die Blattknoten könnten Methoden sein.

2.2 VERWANDTE ARBEITEN

Natürlich gibt es viele Arbeiten und sogar ganze Konferenzen zum Thema Softwarevisualisierung. Aber obwohl ich die Literatur durchsucht habe, sind mir keine vergleichbaren Arbeiten bekannt, also Arbeiten, die sich speziell mit der Darstellung „holistischer“ Metriken im Kontext der Softwarevisualisierung beschäftigen.

3 KONZEPT

Im folgenden Abschnitt wird erläutert, wie verschiedene Problemstellungen das Konzept für die Implementierung beeinflusst haben.

3.1 NUTZERGRUPPEN

Zunächst wird beschrieben, welche Personen als Nutzer des Frameworks erwartet wurden.

3.1.1 *SEE-Nutzer - Softwareentwickler und andere Beteiligte*

Die „Endnutzer“ des Frameworks sind potenziell die, die auch sonst SEE benutzen (würden). Somit gehören zu dieser Benutzergruppe (vermutlich):

- Softwareentwickler. Diese dürften mit der Software-Domäne so vertraut sein, dass sie mit der Bedienung des Frameworks keine Probleme haben dürften, wenn man im User Interface keine SEE-Spezifischen „Vokabeln“ einbaut. Allgemeine Begriffe aus der Informatik sollten ihnen einigermaßen geläufig sein.
- Stakeholder. Damit sind Personen gemeint, die sonst irgendwie am Softwareentwicklungsprozess beteiligt sind, also zum Beispiel der Kunde, der die Entwicklung in Auftrag gibt. Auch von diesen Nutzern darf man erwarten, dass sie in der Lage sind, eine „durchschnittlich komplexe Software“ zu bedienen. Um die Nutzung für sie nicht allzu schwer zu machen, wäre es gut, möglichst wenige Begriffe aus der Informatik einzubauen.

3.1.2 *SEE-Entwickler*

Andere Entwickler, die ebenfalls an SEE arbeiten, könnten das Framework um neue Metriken erweitern wollen. Diese Personen sind zwar nicht Nutzer des Frameworks im eigentlichen Sinne. Da dieser Fall aber für die Implementierung relevant war, werden die SEE-Entwickler hier dennoch als Nutzergruppe angeführt.

3.2 VORGABEN

Diese Vorgaben wurden aus dem Exposé zu dieser Arbeit und aus der Kommunikation mit dem Erstgutachter abgeleitet, sie stehen also fest und werden daher hier nicht unbedingt diskutiert. Dies sind auch Vorgaben, die im Laufe der Implementierungsphase festgelegt wurden.

1. Die Implementierung sollte anderen Programmierern eine Schnittstelle bieten, um mit möglichst geringem Aufwand neue Metriken zu implementieren und zu visualisieren.
2. Die Auswahl an angezeigten Metriken sollte konfigurierbar sein.
3. Der Nutzer sollte wählen können, mit welchem Widget eine Metrik dargestellt wird.
4. Der Nutzer sollte die Widgets frei auf dem Board platzieren können. (Was mit dem Board gemeint ist, lässt sich in 4.1 nachvollziehen)
5. Die Bedienung des Frameworks sollte aus dem Spiel heraus erfolgen.
    a) Das bedeutet, dass die Nutzer die jeweiligen Komponenten nicht über den *Unity*-Editor hinzufügen und konfigurieren müssen. Stattdessen musste SEE um eine Benutzerschnittstelle erweitert werden, die die Bedienung des Frameworks aus dem Spiel heraus ermöglicht.

Unity: *Unity ist eine Videospiel-Engine, also eine Software, die als Rahmenwerk bei der Spieleentwicklung genutzt werden kann.*

3.3 VERSCHIEDENE PROBLEME

In diesem Abschnitt werden wir uns die verschiedenen Problemstellungen anschauen, die für die Implementierung relevant waren.

3.3.1 *Platzierung der Widgets*

Es stellte sich die Frage, wo in der *Szene* die Widgets angebracht werden sollten. Es war sinnvoll, die Widgets auf einer zweidimensionalen Fläche anzubringen, weil das Plugin, das für diese Bachelorarbeit vorgegeben war, fast nur *UI*-Elemente enthält, die zweidimensional sind.

Eine Idee war, die graphischen Elemente seitlich am Tisch anzubringen, auf dem sich die Code City befindet. Diese Idee wurde jedoch verworfen, weil eine Anforderung an die Implementierung ist, dass die Nutzer für andere sichtbar auf die Metrik-Objekte zeigen können. Das hätte aber vorausgesetzt, dass die Teilnehmer alle an der gleichen Seite des Tisches stehen. Gegen diese Umsetzung sprach außerdem, dass man immer mit einem recht steilen Winkel auf die Metriken schaut (von oben nach unten).

Szene: *Das ist in der Unity-Engine sozusagen **eine** Spielwelt. Die Szene von SEE, in der ich für meine Bachelorarbeit gearbeitet habe, besteht hauptsächlich aus einem Boden, dem Avatar des Spielers und einem Tisch, auf dem man eine Code City laden und betrachten kann.*

UI: *Steht für User Interface. Das ist eine Oberfläche, über welche der Nutzer einer Software mit dieser interagiert.*

Letztlich fiel die Entscheidung darauf, die Widgets auf einer eigenen Anzeigetafel anzubringen.

3.3.2 *Mehrfache Schleifendurchläufe*

Potenziell müssen etliche Metriken berechnet werden. Bei vielen Metriken kann es dabei nötig sein, beispielsweise durch alle Knoten der Szene zu iterieren. Dies wiederum würde die Rechenzeit sehr erhöhen, wenn die Evolution einer Code City abgespielt wird - dann wird bei jedem Schritt die Code City neu geladen und die Metriken müssen neu berechnet werden. Es war daher eigentlich naheliegend, die hierfür nötigen Schleifen in einer Schleife zusammenzufassen. So könnte man einiges an Laufzeit sparen.

Allerdings sollte das Framework für andere Programmierer auch leicht erweiterbar sein. Eine immer größer werdende Methode, die von den Programmierern erweitert werden muss, war also nicht geeignet. Stattdessen wurde hier auf die Optimierung im Sinne der Rechenzeit verzichtet, alle Metriken wurden separat implementiert.

3.3.3 *Position der Widgets auf den Anzeigetafeln*

Nun war noch die Frage der Platzierung und Skalierung der Widgets zu klären. Es folgen einige mögliche Ansätze:

- Der Nutzer wählt selbst die x- und y-Koordinaten sowie die Größe der Widgets auf der Tafel
- Es werden mehrere vordefinierte Positionen angeboten, die der Nutzer mit Widgets belegen kann
  - Man könnte die Positionen in einem gleichmäßigen Raster anlegen, wobei alle Widgets gleich groß sind
  - Man könnte die Positionen weniger regelmäßig anbieten und die Widgets an manchen Positionen größer skalieren als an anderen. So könnten Widgets, deren angezeigter Wert öfter abgelesen wird, größer dargestellt werden.

Später ergab sich die Anforderung, dass die Position der Widgets frei wählbar sein muss. Das war technisch vermutlich schwieriger umzusetzen. Einfacher wäre es wahrscheinlich gewesen, mit mehreren vordefinierten Positionen zu arbeiten. Allerdings ist die manuelle Positionierung für die Nutzer von SEE vermutlich die beste Option.

4 UMSETZUNG

In diesem Kapitel soll es darum gehen, wie das zuvor beschriebene Konzept mit Hilfe von Unity und C# implementiert wurde.

C#: Das ist eine Programmiersprache, die unter anderem innerhalb der Unity-Engine für eigene Skripte verwendet werden kann.

4.1 METRICS BOARDS

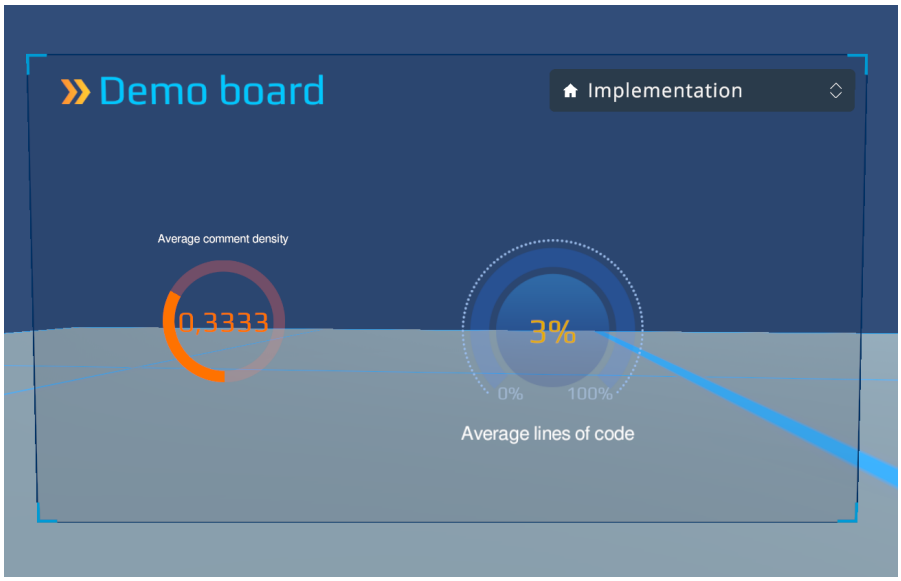

Abbildung 4.1: Beispiel für ein Metrics Board

Mittelpunkt des Frameworks sind die Metrics Boards. Diese können als leere Leinwände gesehen werden, die Platz für verschiedene Widgets bieten und diese verwalten. Die Widgets wiederum bestehen aus zwei „Teilen“: Zum einen die dynamische Grafik, wie zum Beispiel ein buntes Tacho, und eine Schnittstelle, um diese Grafik verschiedene Werte anzeigen zu lassen. Zum anderen die Metrik, die eine Schnittstelle bietet, um ihren aktuellen Wert in Bezug auf eine Code City abzufragen. Diese beiden Teile ruft das Metrics Board bei Bedarf auf, um auf den verwalteten Widgets sinnvolle Werte anzuzeigen. Ferner kann für jedes Metrics Board ausgewählt werden, auf welche Code City dieses sich bezieht.

Ein Beispiel für ein Metrics Board findet sich in Abbildung 4.1. Das dort abgebildete Board mit dem Titel „Demo board“ enthält zwei Widgets und hat (so wie alle Metrics Boards) oben rechts eine

Dropdown-Liste, mit der die Code City ausgewählt werden kann, deren Metriken angezeigt werden.

4.2 MENÜ

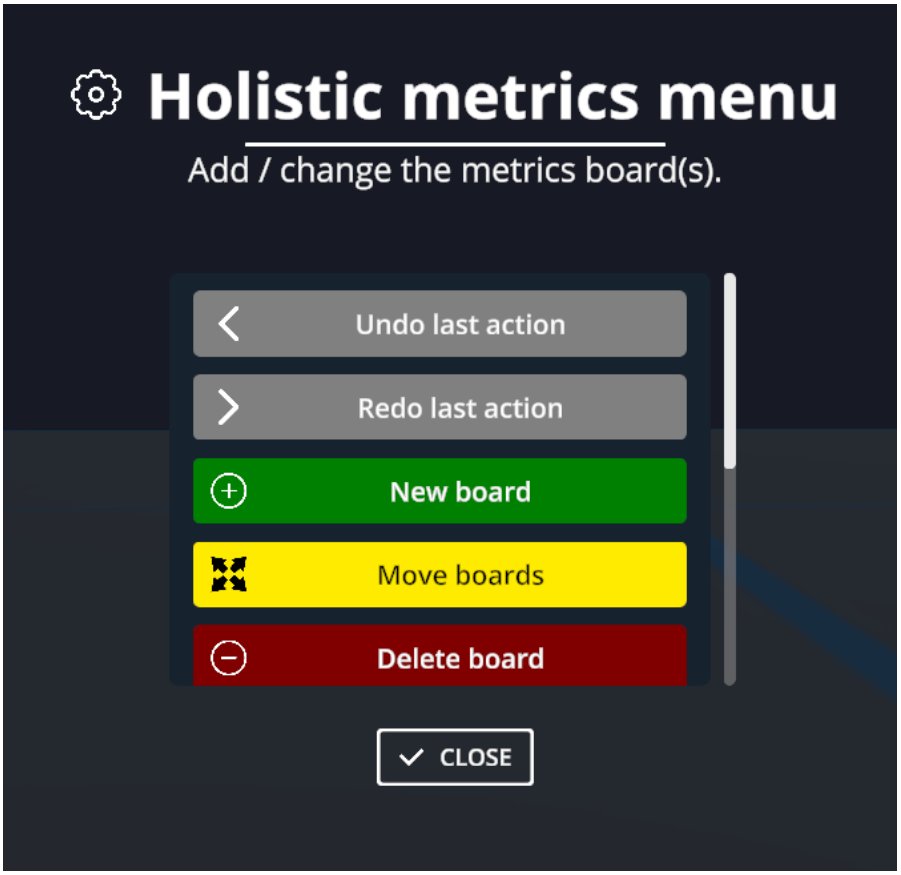

Abbildung 4.2: Das Menü, mit dem das Framework bedient wird. Zu sehen sind nur die ersten fünf Einträge (innerhalb des Menüs kann gescrollt werden).

Für die einfache Bedienung des Frameworks existiert ein Menü. Dieses kann vom Nutzer per Tastendruck geöffnet werden und bietet folgende Optionen:

- Ein neues, leeres Metrics Board erstellen
- Metrics Boards verschieben
  - Dabei werden die Boards auf dem Boden verschoben, wobei auch die Rotation angepasst wird
- Metrics Boards löschen
- Widgets erstellen (dabei kann frei gewählt werden, welche Metrik angezeigt werden soll und auch, mit welcher Grafik diese visualisiert werden soll)

- Widgets verschieben (auf dem Metrics Board)
- Widgets löschen
- Die vollständige Konfiguration eines Metrics Boards in einer Datei speichern
- Ein Metrics Board aus einer Konfigurationsdatei laden
- Alle Optionen können außerdem rückgängig gemacht und wiederhergestellt werden (mit Außnahme des Speicherns eines Metrics Boards in einer Datei)

Somit lässt sich das Framework relativ benutzerfreundlich aus dem Spiel heraus konfigurieren. Beim Anklicken einer der Optionen aus dem Menü wird teilweise auch ein kleiner Hinweis angezeigt, wie die jeweilige Funktion zu benutzen ist. Außerdem öffnet sich bei manchen Optionen ein Dialog, in dem vom Nutzer noch weitere Informationen eingegeben werden müssen, wie zum Beispiel der Titel des neuen Boards, wenn ein neues Board erstellt werden soll.

Es soll noch angemerkt werden, dass das Menü von einem bereits in SEE vorhandenen „Menü-Framework“ Gebrauch macht. Da dieses auch für andere Menüs in SEE genutzt wurde, war es sinnvoll, dies auch für das neue Menü zu nutzen. So müssen Nutzer sich im Kopf nicht auf ein neues Menü einstellen, sondern haben eine gewohnte Darstellung vor sich.

4.3 ERWEITERBARKEIT

Das Framework ist vor allem bei der Sammlung der Metriken sehr einfach erweiterbar. Andere Entwickler können hier einfach eine neue Klasse erstellen, die von der `Metric`-Klasse erbt. Dabei müssen sie nur eine Methode implementieren, deren vorgegeben ist. Was in der Methode berechnet wird, steht den Entwicklern dabei vollständig offen. Auch muss die Klasse tatsächlich nur implementiert werden, sie wird dann vom Framework automatisch eingebunden und dem Nutzer in SEE zur Auswahl angeboten.

Die zu implementierende Methode bekommt als Eingabe eine Instanz der Klasse `SEECity`, also einer Code City im Spiel. Welche Code City das ist, wird vom Metrics Board verwaltet (und kann vom Nutzer eingestellt werden).

Als Rückgabe der Methode wird eine Instanz vom eigens implementierten Typ `MetricValue` erwartet. Diese lässt sich einfach erstellen und enthält entweder einen einzelnen Metrik-Wert oder eine Sammlung von Metrik-Werten, außerdem Ober- und Untergrenze der Metrik-Werte (der schlechteste und der beste Wert, den man noch beachtet) und den Namen der Metrik.

5 EVALUATION

In diesem Abschnitt wird beschrieben, wie das Ergebnis der Implementierung evaluiert wurde. Zum Zwecke der Evaluation wurde eine Studie durchgeführt, bei der die Probanden einige Aufgaben bearbeiten und einen Fragebogen ausfüllen mussten.

5.1 FORSCHUNGSFRAGE

Die Forschungsfrage, die diese Arbeit beantworten sollte, lautet: *Wie lassen sich Metriken zu Eigenschaften ganzer Code-Cities geeignet in SEE visualisieren?* Diese wurde in die drei Aspekte Usability, Design und Mehrwert aufgeteilt, wobei hier der Mehrwert jeweils getrennt in Form von Korrektheit und Zeitersparnis betrachtet wird. Diese *vier* Aspekte sollen nun betrachtet werden.

5.1.1 *Usability*

Das Framework sollte eine gute Usability (deutsch: Gebrauchstauglichkeit) haben. Hierzu sollte ein Fragebogen ausgefüllt werden, zum Beispiel der SUS-Fragebogen. Die Frage lautete dann: Nehmen die Nutzer das Framework in Bezug auf Usability im Durchschnitt eher positiv oder negativ war?

5.1.2 *Design*

Die Darstellung sollte ansprechend sein, um den Nutzer zu motivieren. Hierzu sollte ein Fragebogen zur Hedonic Usability ausgefüllt werden. Die Frage lautete dann: Nehmen die Nutzer das Framework in Bezug auf ein ansprechendes Design im Durchschnitt eher positiv oder negativ wahr?

5.1.3 *Korrektheit*

Es wurde auch ermittelt, ob die Korrektheit der Antworten der Probanden mithilfe des Frameworks gesteigert werden konnte, oder ob die Antworten vielleicht sogar weniger korrekt waren.

HYPOTHESE Wir formulieren nun diese Fragestellung in Form einer Nullhypothese und einer alternativen Hypothese. Dabei bezeichnen

wir die Korrektheit ohne Framework als c_o, die Korrektheit mit dem Framework als c_f. Dann lautet die Nullhypothese

$$H_{0_c} :\Leftrightarrow c_o \geq c_f$$

Die alternative Hypothese lautet

$$H_{1_c} :\Leftrightarrow c_o < c_f$$

5.1.4 *Zeitersparnis*

Die Frage, ob der Nutzer mit dem Framework Zeit sparen kann, kann man sofort beantworten.

Angenommen, der Nutzer möchte die Anzahl an Knoten in der Code City feststellen und angenommen, diese Metrik wurde im Framework bereits implementiert.

Nun hängt die Zeitersparnis davon ab, wie groß die betrachtete Code City ist. Ist die Code City groß in Bezug auf die Anzahl Knoten beziehungsweise Kanten, ist auch die Zeitersparnis durch das Framework groß. Denn das manuelle Zählen dauert tendenziell länger, je mehr Elemente gezählt werden, während der Zeitaufwand für das Anzeigen und Ablesen mithilfe des Frameworks unabhängig von der Größe der Code City ist.

Weil der Aufwand hierfür nicht all zu groß sein sollte, werden wir dies im Rahmen der Studie dennoch evaluieren. Wir können die erhaltenen Daten auch nutzen, um zu erfahren, *wie viel* schneller Nutzer mit dem Framework Aufgaben bearbeiten können.

Die Frage lautet dann: Brauchen Nutzer im Durchschnitt weniger Zeit für ihre Aufgaben, wenn sie das Framework nutzen?

HYPOTHESE UND NULLHYPOTHESE Nun formulieren wir die Nullhypothese und die alternative Hypothese. Dabei nennen wir die Bearbeitungszeit ohne Framework t_o, die Bearbeitungszeit mit Framework: t_f. Nun definieren wir die Nullhypothese als

$$H_{0_t} :\Leftrightarrow t_o \leq t_f$$

Die alternative Hypothese definieren wir als

$$H_{1_t} :\Leftrightarrow t_o > t_f$$

5.2 VERSUCHSAUFBAU

Den Probanden wurde ein Link zu einem Online-Fragebogen zugesendet. In diesem Fragebogen enthalten war auch eine Anleitung, in der die Steuerung und Bedienung von SEE und vom Framework erklärt wurde. Auch gab es Links zu zwei Videos, in denen die Code Cities

und das Framework erklärt wurden, in der Hoffnung, dass die Probanden es weniger anstrengend finden, die Videos anzuschauen, als beispielsweise eine textuelle Anleitung zu lesen (beide Videos befinden sich auch im Anhang dieser Arbeit). Alternativ waren aber auch zwei Dokumente verlinkt (ebenfalls im Anhang dieser Arbeit vorhanden), in denen ebenfalls SEE und das Framework erklärt wurden. Diese Dokumente wurden aufgrund des Feedbacks aus der Pilotstudie erstellt. In der Anleitung befand sich auch ein Download-Link für SEE, wobei die notwendigen Code Cities hier bereits vorhanden waren (diese Version von SEE befindet sich auch im Anhang dieser Arbeit).

Nachdem die Probanden die Anleitung gelesen hatten, kam als erstes eine Aufgabe, mit der sichergestellt werden sollte, dass die Probanden die Anleitung verstanden hatten. Die Probanden mussten nun folgende Frage beantworten: „Was ist der Wert der Number of Leaf Nodes-Metrik für die Code City namens City Task 0?“. Um diese Frage zu beantworten, mussten die Probanden eine neue Metrik-Anzeigetafel in der Spielwelt platzieren, für diese Anzeigetafel die richtige Software-Stadt auswählen, deren Werte darauf angezeigt werden sollen, ein Widget konfigurieren und auf der Tafel platzieren, welches die Number of Leaf Nodes anzeigt. Dann konnte der korrekte Wert einfach abgelesen und anschließend in der Umfrage engetippt werden. Als zweites mussten die Probanden noch die „Number of Edges“ ermitteln. Das sollten sie tun, indem sie manuell zählen. Die Probanden konnten im Fragebogen nicht ohne Weiteres fortfahren, bevor sie die **richtigen** Werte eingetragen hatten. Dies war bei den nächsten Aufgaben anders, die Probanden konnten hier nicht erfahren, ob sie die richtigen Werte eingetragen hatten. Das erlaubte es, im Nachhinein die Korrektheit der Antworten in die Auswertung einzubeziehen. Hätte man die Probanden erst fortfahren lassen, wenn sie die richtige Antwort eingegeben haben, wäre das nicht der Fall.

Nun gab es vier Aufgaben zu bearbeiten, zu denen je die Bearbeitungszeit zu messen war. Davon waren zwei Aufgaben **ohne** das Framework zu bearbeiten, zwei Aufgaben **mit** dem Framework. Um Lerneffekte auszuschließen, wurden die Probanden an dieser Stelle in zwei Gruppen aufgeteilt, von denen eine zuerst die Aufgaben mit dem Framework bearbeitete, während die andere zuerst die Aufgaben ohne das Framework bearbeitete. Anhand der Bearbeitungszeiten sollte dann erkennbar sein, dass es sehr viel einfacher ist, die entsprechenden Werte mit dem Framework herauszufinden. Wichtig war hierbei, dass der Zeitgewinn durch das Framework davon abhängt, wie groß die Code Cities sind, die betrachtet werden. Wenn die Aufgabe zum Beispiel ist, die Knoten zu zählen, und die Code City nur einen Knoten enthält, ist es wahrscheinlich schneller, einfach die Code City anzusehen, als ein Metrics Board zu erstellen und zu konfigurieren, um anschließend den Wert abzulesen (wobei die Metrics Boards natürlich nicht so konzipiert sind, dass man jedes mal ein neues anlegt, wenn man eine Metrik erfah-

ren will). Wenn die gleiche Aufgabe hingegen für eine Code City mit hunderten Knoten zu bearbeiten ist, wäre die Zeitersparnis durch das Framework wahrscheinlich extrem groß. Außerdem wäre bei großen Code Cities die Fehlerquote beim manuellen Zählen sicherlich größer.

Nach der Bearbeitung jeder Aufgabe war die Antwort in den Fragebogen einzutragen, ebenso die Bearbeitungszeit. Die genauen Aufgabenstellungen werden im Kapitel 5.5 genau beschrieben.

Nachdem alle Aufgaben bearbeitet wurden, sollten die Probanden noch einige Fragen zur Usability (Pragmatic Usability sowie Hedonic Usability) beantworten.

5.2.1 *Unabhängige Variablen*

Jetzt sollen möglichst vollständig die unabhängigen Variablen aufgezählt werden, also solche, die das Ergebnis der Studie beeinflussen, die allerdings durch die Probanden nicht beeinflusst wurden.

NUTZUNG DES FRAMEWORKS Dies ist die Variable, die gezielt variiert wurde, um zu sehen, ob das Framework die Korrektheit steigert und die Zeit reduziert, mit der Aufgaben bearbeitet wurden. Dementsprechend gab es Aufgaben, bei denen das Framework verwendet werden sollte, und Aufgaben, bei denen es nicht verwendet werden sollte.

ALTER Das Alter wurde zwar erhoben, aber nicht gezielt variiert.

GESCHLECHT Das Geschlecht der Probanden wurde ebenfalls erhoben, aber nicht gezielt variiert. Alter und Geschlecht der Probanden könnten aber durchaus Einfluss auf das Ergebnis der Studie gehabt haben.

ERFAHRUNG MIT SEE Die Erfahrung mit SEE könnte eine große Rolle gespielt haben. Um ein Beispiel zu nennen: Es könnte sein, dass langjährige SEE-Entwickler einen Vorteil bei der Bedienung von SEE ohne Framework gehabt hätten und die entsprechenden Aufgaben schneller bearbeitet hätten, während sie durch das neue Framework keinen viel größeren Zeitvorteil gehabt hätten, als andere Probanden. Deshalb war es sinnvoll, auch diese Variable zu erfassen.

ERFAHRUNG MIT 3D-VIDEOSPIELEN Da SEE selbst ein dreidimensionales Videospiel ist, war es auf jeden Fall nützlich, Erfahrung mit anderen Spielen zu haben. Die Probanden, die bereits solche Erfahrungen hatten, konnten sich vermutlich schneller in SEE zurechtfinden und die Aufgaben teilweise schneller bearbeiten.

GRÖSSE DER CODE CITY Die Größe der Code City in der Anzahl an Knoten und Kanten hat natürlich beeinflusst, wie lange die Probanden gebraucht haben, um die Anzahl an Knoten und Kanten zu zählen.

5.2.2 *Abhängige Variablen*

Nun werden vollständig alle abhängigen Variablen angeführt, die in der Studie betrachtet wurden.

BEARBEITUNGSZEIT MIT FRAMEWORK Dies entspricht der Variable t_f aus den Hypothesen. Für diese Variable bekommen wir Werte aus vier verschiedenen Aufgaben: Die ersten beiden Aufgaben (Einführungsaufgaben nicht mitgezählt) wurden von Gruppe A **mit** dem Framework bearbeitet, die letzten beiden Aufgaben wurden von Gruppe B mit dem Framework bearbeitet. Gruppe A wurde bei der zweiten Aufgabe, Gruppe B bei der vierten Aufgabe aufgefordert, das Board aus der jeweils vorherigen Aufgabe wiederzuverwenden. Somit würden wir bei diesen Aufgaben eine Zeitersparnis erwarten. Auch könnte sich hier ein Lerneffekt eingestellt haben.

BEARBEITUNGSZEIT OHNE FRAMEWORK Dies entspricht der Variable t_o. Ähnlich wie bei t_f erhalten wir die Werte dieser Variablen auch aus vier verschiedenen Aufgaben.

KORREKTHEIT MIT FRAMEWORK Dies entspricht der Variable c_f. Diese Variable hängt davon ab, wie gut die Probanden das Framework verstanden haben, aber auch von der Erfahrung mit SEE und 3D-Videospielen.

KORREKTHEIT OHNE FRAMEWORK Dies entspricht der Variable c_o.

SYSTEM USABILITY SCALE Hier werden wir pro Proband einen Wert zwischen 0 (am schlechtesten) und 100 (am besten) erhalten, der uns verrät, wie die Probanden jeweils die Nutzbarkeit des Frameworks einschätzen.

MECUE 2.0 Wie schätzen die Nutzer das Framework ein? Hier erhalten wir ebenfalls einen Score für jeden Probanden, indem die sieben Antwortmöglichkeiten auf die Zahlen 1 bis 7 abgebildet und dann verrechnet werden.

5.3 FRAGEBOGEN

Der Fragebogen sollte über das Internet ausgefüllt werden. Dafür wurden mehrere Tools angeschaut. Ein Tool ist Surveymonkey. Das Tool Surveymonkey (`surveymonkey.com`) ist nicht geeignet, da es nur bis zu 10 Fragen erlaubt. Auch in Betracht gezogen wurde `limesurvey.`

org. Dies ist auch nicht geeignet, da es Geld kostet. Letztlich musste sich zwischen KoBoToolbox und Google Forms entschieden werden. Nachdem beide Tools ausprobiert wurden, fiel die Entscheidung auf KoBoToolbox, weil es damit möglich ist, die Bearbeitungszeit von Aufgaben zu messen (indirekt).

Der Fragebogen setzte sich letztlich aus mehreren Fragebögen / Teilen zusammen. Im Folgenden wird erläutert, wie diese gewählt wurden.

5.3.1 *Demographischer Abschnitt*

Diese Daten sind unabhängige Variablen, bei denen nicht klar war, ob sie einen Einfluss auf das Ergebnis haben. Daher wurden sie erfragt, falls sie bei der späteren Auswertung berücksichtigt werden sollten. Die Erfahrung mit 3D-Videospielen wurde mit einer Likert-Skala mit fünf Punkten erfasst.

5.3.2 *Abschnitt zur Bearbeitungszeit und Korrektheit*

In diesem Abschnitt wurden den Probanden bestimmte Aufgaben gestellt. Die Aufgaben kamen dabei nicht aus einer anderen wissenschaftlichen Arbeit - es waren einfach Aufgaben, zu denen die Probanden jeweils eine Lösung eingeben mussten. Nachdem genug Daten gesammelt wurden, konnten die Bearbeitungszeit und Korrektheit der Fragen ermittelt werden. Welche Aufgaben den Probanden gestellt wurden, kann in Kapitel 5.5 ersehen werden.

5.3.3 *Abschnitt zur Usability*

Hier wurde zwischen mehreren Fragebögen aus der Wissenschaft abgewogen. Im Folgenden werden kurz einige Alternativen verglichen.

ISONORM 9241/10 Dieser Fragebogen ist einer der zwei wichtigsten im deutschsprachigen Raum, die sich auf die ISO Norm 9241/10 beziehen. Er nimmt etwa zehn Minuten in Anspruch (frei zitiert aus Figl (2009)). Da er jedoch 35 Fragen beinhaltet, ist er zu umfangreich.

ISOMETRICS Dieser Fragebogen ist ebenfalls einer der zwei wichtigsten im deutschsprachigen Raum, die sich auf die ISO Norm 9241/10 beziehen (frei zitiert aus Figl (2009)). Er besteht allerdings aus 75 Fragen und wird deshalb ebenfalls nicht ausgewählt.

TECHNOLOGIEAKZEPTANZMODELL Das Technologieakzeptanzmodell wurde in einer Arbeit von Davis (1985) entwickelt. Dieses Modell stellt anscheinend keinen konkreten Fragebogen zur Verfügung, dieser müsste wohl anhand des Modells erst erarbeitet werden. Da dieser Aufwand zu hoch erscheint, kommt dieses Modell nicht in die engere

Auswahl. Außerdem hätte man, nachdem man den konkreten Fragebogen selbst (anhand des Modells) erstellt hätte, keinen Fragebogen, der direkt bestätigt wurde (andere Fragebögen wurden bereits in der Wissenschaft überprüft).

SYSTEM USABILITY SCALE Die System Usability Scale entstammt einem Artikel von Brooke (1996). Dieser Fragebogen enthält nur zehn Fragen und ist sehr verbreitet. Daher fiel die Entscheidung darauf, mit diesem Fragebogen die Usability des Frameworks zu erheben.

Der gewählte Fragebogen „System Usability Scale“ ist im Original nicht auf Deutsch verfasst worden, es gibt jedoch eine Übersetzung ins Deutsche (Rummel und Ruegenhagen (2015)). Diese wurde für den Fragebogen übernommen, wie er auch unten aufgeführt ist.

5.3.4 *Abschnitt zur Ästhetik*

Hier wurde nochmals ein separater Fragebogen verwendet, um die „Hedonic Usability“ des Frameworks zu erheben.

ATTRAKDIFF Der AttrakDiff-Fragebogen enthält Fragen zur Usability **und** zum Design. Da die Usability aber schon mit dem System Usability Scale-Fragebogen abgedeckt wurde, würde dies die Anzahl an Fragen unnötig erhöhen. Ich konnte auch nichts darüber finden, dass der Fragebogen modular aufgebaut ist, so dass man einen Teil der Fragen hätte weglassen können. Daher kam dieser Fragebogen nicht in Frage.

USER EXPERIENCE QUESTIONNAIRE Dieser Fragebogen wäre vielleicht geeignet gewesen, schien auf den ersten Blick jedoch nicht ganz unkompliziert zu benutzen zu sein. Um den (zeitlichen) Aufwand nicht weiter zu erhöhen, wurde sich gegen diesen Fragebogen entschieden. Auch bei diesem Fragebogen war nicht ohne weiteres zu sehen, wie man die Fragen in Kategorien trennen könnte, um einige Fragen wegzulassen, die nicht benötigt wurden.

MECUE 2.0 Dieser Fragebogen erschien sehr geeignet. Er ist auf Englisch und Deutsch verfügbar. Er enthält zwar auch einige Fragen zur Nützlichkeit und Benutzbarkeit, die nicht benötigt wurden. Er ist aber auch modular aufgebaut (Minge und Riedel (2013)), und alle fünf Module wurden separat validiert. Um die Anzahl an Fragen möglichst gering zu halten, fiel die Entscheidung darauf, nur das zweite Modul zu verwenden, welches nicht-aufgabenbezogene Produkttwahrnehmungen erfasst. Somit kamen zum Fragebogen neun weitere Fragen hinzu.

5.3.5 *Auflistung der Fragen*

Es folgt nun eine Auflistung der Fragen in der Reihenfolge, wie sie auch im Fragebogen standen. Hier ist allerdings nicht immer die exakte Formulierung vorhanden. Diese ist im Anhang in Form von zwei Excel-Dateien vorhanden (eine für Gruppe A, eine für Gruppe B). Außerdem wurden die Probanden jeweils vor Beginn einer Aufgabe gefragt, ob sie nun bereit sind, diese zu bearbeiten. Das war aus technischen Gründen nötig, um die Bearbeitungszeit zu erfassen.

1. Demographischer Teil:
   a) Alter
   b) Geschlecht
   c) Beruf (falls Student: Mit Studiengang)
   d) Erfahrung mit 3D-Videospielen (könnte bei Bedienung helfen, wie in Galperin (2021))
   e) Hast du bereits Erfahrung mit SEE?
2. Übungsaufgaben:
   a) Wie viele Blattknoten hat die „City Task 0“?
      - Mit Framework
      - Mit dieser Frage wollten wir nur sicherstellen, dass der Proband die Anleitung verstanden hat und in der Lage ist, das Framework zu bedienen.
   b) Wie viele Kanten hat die „City Task 0“?
      - Ohne Framework
      - Damit sollte sichergestellt werden, dass die Probanden die kommenden Aufgaben bearbeiten können.
3. Teil zum Mehrwert:
   a) Wie viele Blattknoten hat die „City Task 1 - 2“?
      - Gruppe A: Mit Framework
      - Gruppe B: Ohne Framework
   b) Wie viele Kanten hat die „City Task 1 - 2“?
      - Gruppe A: Mit Framework
      - Gruppe B: Ohne Framework
   c) Wie viele Blattknoten hat die „City Task 3 - 4“?
      - Gruppe A: Ohne Framework
      - Gruppe B: Mit Framework
   d) Wie viele Kanten hat die „City Task 3 - 4“?
      - Gruppe A: Ohne Framework

- Gruppe B: Mit Framework

4. Abschnitt zur Usability. Alle Fragen wurden mit einer fünfstufigen Likert-Skala beantwortet, die Beschriftung der Antworten war ebenfalls der deutschen Übersetzung des System Usability Scale-Fragebogens entnommen (und wird hier nicht aufgeführt). Vorher kam noch ein Hinweis, dass die Fragen sich nicht auf SEE als ganzes beziehen, sondern nur auf das Framework.
   a) Ich denke, dass ich das System gerne häufig benutzen würde.
   b) Ich fand das System unnötig komplex.
   c) Ich fand das System einfach zu benutzen.
   d) Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.
   e) Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.
   f) Ich denke, das System enthielt zu viele Inkonsistenzen.
   g) Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.
   h) Ich fand das System sehr umständlich zu nutzen.
   i) Ich fühlte mich bei der Benutzung des Systems sehr sicher.
   j) Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.

5. Teil zur Coolness. Die Antwortmöglichkeiten sind im meCUE 2.0-Fragebogen vorgegeben. Dabei gibt es sieben verschiedene Stufen der Zustimmung.
   a) Das Produkt ist kreativ gestaltet.
   b) Das Produkt verleiht mir ein höheres Ansehen.
   c) Ohne das Produkt kann ich nicht leben.
   d) Das Design wirkt attraktiv.
   e) Durch das Produkt werde ich anders wahrgenommen.
   f) Das Produkt ist wie ein Freund für mich.
   g) Das Produkt ist stilvoll.
   h) Wenn ich das Produkt verlieren würde, würde für mich eine Welt zusammenbrechen.
   i) Meine Freunde können wegen des Produkts ruhig neidisch auf mich sein.

Convenience Sampling: *Convenience Sampling bedeutet, dass die Probanden aufgrund ihrer leichten Verfügbarkeit ausgewählt werden.*

5.4 PROBANDEN UND OBJEKTE DER STUDIE

Als Sampling-Methode wurde sich der Einfachheit halber für *Convenience Sampling* entschieden. Viele der Probanden waren informatiknahe Personen. Wenn wir also eine Hypothese bewiesen oder widerlegt haben, können wir dies nicht für alle Bevölkerungsgruppen verallgemeinern. Wir haben dann aber dennoch eine brauchbare Aussage, denn informatiknahe Personen gehören zur Zielgruppe von SEE und somit ist das Ergebnis auch relevant.

Die Zuordnung der Probanden zu den beiden Gruppen erfolgte, indem ich Zettel mit den Namen aller Probanden in eine Urne (eigentlich einen Schlittschuh) legte. Anschließend zog ich 14 Mal einen Namen aus der Urne und ordnete diesen abwechselnd der einen oder der anderen Gruppe zu.

5.5 GESTELLTE AUFGABEN

Das Ziel der Aufgaben für die Probanden war, entweder die Anzahl der Blattknoten einer Code City zu ermitteln, oder die Anzahl der Kanten einer Code City zu ermitteln. Dazu sollte entweder ein Metrics Board verwendet werden, oder es sollte kein Metrics Board verwendet werden (die Probanden hätten also die Knoten beziehungsweise Kanten „von Hand“ zählen müssen). Es gab jeweils eine Code City für die ersten beiden Fragen, eine für die dritte und vierte Frage und eine für die letzten beiden Fragen. Es folgen nun die Aufgabenbeschreibungen, genau wie sie im Fragebogen standen. Dabei werden allerdings nicht sämtliche Anweisungen genannt. Beispielsweise wird die Anweisung, SEE herunterzuladen, oder die Anweisung, SEE zu starten, hier nicht aufgeführt. Die Excel-Dateien für die Fragebögen der beiden Gruppen finden sich aber im Anhang.

1. Aufgaben an der ersten Code City (damit sollte nur sichergestellt werden, dass die Probanden in der Lage sind, die eigentlichen Aufgaben zu bearbeiten):
   a) „Was ist die **Number of Leaf Nodes** der **City Task 0?**“

      Die Probanden wurden darauf hingewiesen, für diese Aufgabe ein Metrics Board zu verwenden.

   b) „Was ist die **Number of Edges** der **City Task 0?**“

      Die Probanden wurden darauf hingewiesen, für diese Aufgabe **kein** Metrics Board zu verwenden.

2. Aufgaben an der zweiten Code City:
   a) „Was ist die Anzahl an Blattknoten (Leaf Nodes) der Code City?“

Die Probanden aus Gruppe A wurden darauf hingewiesen, für diese Aufgabe ein neues Metrics Board zu erstellen. Die Probanden aus Gruppe B sollten das Gegenteil tun.

b) „Ebenfalls am Tisch "Task 1 - 2": Was ist die Anzahl an Kanten (Edges) der Code City? Bitte verwende hierfür ebenfalls das Metrics Board, das du für Aufgabe 1 erstellt hast."

Hier ist mit „Aufgabe 1" die Aufgabe aus 2a gemeint.

Dies ist der Text von Gruppe A. Der Text von Gruppe B lautete: „Ebenfalls am Tisch "Task 1 - 2": Was ist die Anzahl an Kanten (Edges) der Code City? Bitte beantworte dies auch **ohne** ein Metrics Board zu nutzen."

3. Aufgaben an der dritten Code City:

a) „Was ist die Anzahl an Blattknoten (Leaf Nodes) der Code City?"

Die Probanden aus Gruppe A sollten hierfür kein Metrics Board benutzen, die Probanden aus Gruppe B schon.

b) „Ebenfalls an Tisch "Task 3 - 4": Was ist die Anzahl an Kanten (Edges) der Code City? Beantworte dies bitte auch, **ohne** ein Metrics Board zu nutzen."

Dies ist der Text von Gruppe A. Gruppe B hatte folgenden Text: „Ebenfalls an Tisch "Task 3 - 4": Was ist die Anzahl an Kanten (Edges) der Code City? (Benutze hier bitte auch das Metrics Board)"

5.6 ERGEBNISSE

In diesem Kapitel wollen wir die Ergebnisse der Studie betrachten. Die unbearbeiteten Daten sind auch im Anhang dieser Arbeit vorhanden.

5.6.1 *Demographische Daten*

Das Durchschnittsalter der Probanden betrug etwa 27,9 Jahre. Alle Probanden waren männlich, keiner von ihnen gab an, bereits Erfahrung mit SEE zu haben. 7 der 14 Probanden kommen aus dem Feld der Informatik, davon waren wiederum 4 bis 5 Studenten.

Um die durchschnittliche Erfahrung mit 3D-Videospielen zu berechnen, werden die fünf Antwortmöglichkeiten auf die Zahlen 0 bis 4 abgebildet. Es ergibt sich ein Mittelwert von etwa 2,57, also liegt die durchschnittliche Antwort zwischen „Mittelmäßig" und „Viel".

5.6.2 *Mehrwert*

KORREKTHEIT Bei der ersten und zweiten Aufgabe sollte Gruppe A das Framework verwenden, Gruppe B nicht. Bei der dritten und vierten Aufgabe sollte Gruppe B das Framework verwenden, Gruppe A hingegen nicht. Wir werden die Standardabweichung der Gruppen pro Aufgabe betrachten, wobei wir nicht mit dem Durchschnittswert vergleichen, sondern mit dem korrekten Wert. Außerdem sind die Werte größtenteils gerundet. Die Formel für die Berechnung lautet:

$$\sqrt{\frac{\sum_{i=1}^{7}(x_i - k)^2}{7}}$$

Dabei ist k der korrekte Wert der jeweiligen Aufgabe. x_i ist jeweils die Antwort eines Probanden (beide Gruppen hatten 7 Probanden). Wir sehen (in Abbildung 5.1), dass bei der ersten Aufgabe die Gruppe

Aufgabe	Gruppe A	Gruppe B
1	10,96	33,74
2	4,54	1,36
3	12,26	0
4	6,76	0

Abbildung 5.1: Standardabweichung der beiden Gruppen pro Aufgabe.

mit dem Framework besser abgeschnitten hat, als die Gruppe **ohne** Framework. Bei der zweiten Aufgabe hingegen hat sie sogar schlechter abgeschnitten. Bei der dritten und vierten Aufgabe sehen wir, dass die Gruppe **mit** Framework keine Fehler gemacht hat, während die Gruppe **ohne** Framework durchaus Fehler gemacht hat.

ZEITERSPARNIS Gruppe A benötigte für die erste Aufgabe im Durchschnitt 02:33 Minuten, Gruppe B nur 02:09 Minuten. Somit hat die Bearbeitung mit dem Framework sogar etwas länger gedauert. Bei der zweiten Aufgabe brauchte Gruppe A im Schnitt 00:51 Minuten, Gruppe B 00:49 Minuten. Auch hier gibt es also keine Zeitersparnis durch das Framework. Nun zur dritten und vierten Aufgabe. Gruppe B brauchte für die dritte Aufgabe durchschnittlich 02:16 Minuten, Gruppe A nur 01:15 Minuten. Auch hier war die Gruppe **ohne** Framework schneller, sogar sehr deutlich. Bei der letzten Aufgabe hingegen benötigte Gruppe B nur 00:26 Minuten, während Gruppe A 01:04 Minuten brauchte. Nur bei dieser Aufgabe war also die Gruppe **mit** dem Framework schneller. Aufgrund dieser Erkenntnisse müsste man sagen, dass die Nullhypothese H_{0_t} gilt und die alternative Hypothese H_{1_t} zu verwerfen ist, dass also das Framework keine Zeitersparnis bringt. In Abbildung 5.2 sind alle Zeiten noch einmal übersichtlicher aufgelistet.

Aufgabe	Gruppe A	Gruppe B
1	02:33	02:09
2	00:51	00:49
3	01:15	02:16
4	01:04	00:26

Abbildung 5.2: Durchschnittliche Bearbeitungszeit der beiden Gruppen pro Aufgabe in Minuten.

5.6.3 *Usability*

Die Ergebnisse des Fragebogens für die SUS (System Usability Scale) sind durchschnittlich ausgefallen. Hier zunächst eine Grafik, die die SUS-Scores aller Teilnehmer zeigt: Wir erhalten als Mittelwert rund

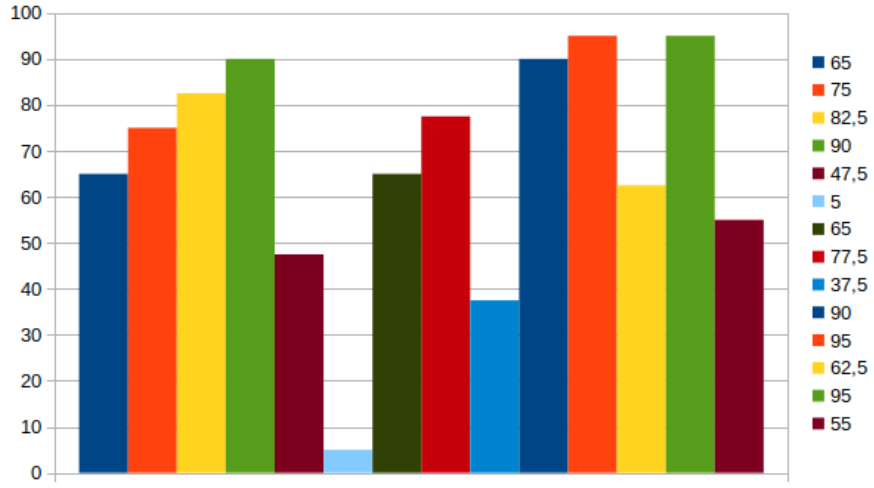

Abbildung 5.3: SUS-Scores aller Teilnehmer

67, 321. Damit liegt das Framework gemäß Bangor u. a. (2009) zwischen „OK“ und „gut“. Da laut selbigem Artikel (S. 120) traditionell ein Produkt ab einem SUS-Score von 70 „bestanden“ hat, könnte man das Framework vielleicht als nicht benutzerfreundlich einstufen.

5.6.4 *Design*

Nun zum Ergebnis des zweiten verwendeten Fragebogens, des meCUE 2.0. Die Probanden hatten hier sieben Antwortmöglichkeiten, von „lehne völlig ab“ bis „stimme völlig zu“. Für die Auswertung mussten zunächst die Antworten der Probanden auf die Zahlen 1 bis 7 abgebildet werden, wobei 1 der Antwort „lehne völlig ab“ entspricht, 7 der Antwort „stimme völlig zu“, die übrigen Werte analog. Mit dem gewählten Fragebogen haben wir die visuelle Ästhetik, den Status und die Bindung des Frameworks in Bezug auf die Probanden erhoben, wobei wir uns nur

für die visuelle Ästhetik interessieren. Der Mittelwert der Antworten der Probanden zu den Fragen in Bezug auf die Ästhetik ist 4, 71 (von 7).

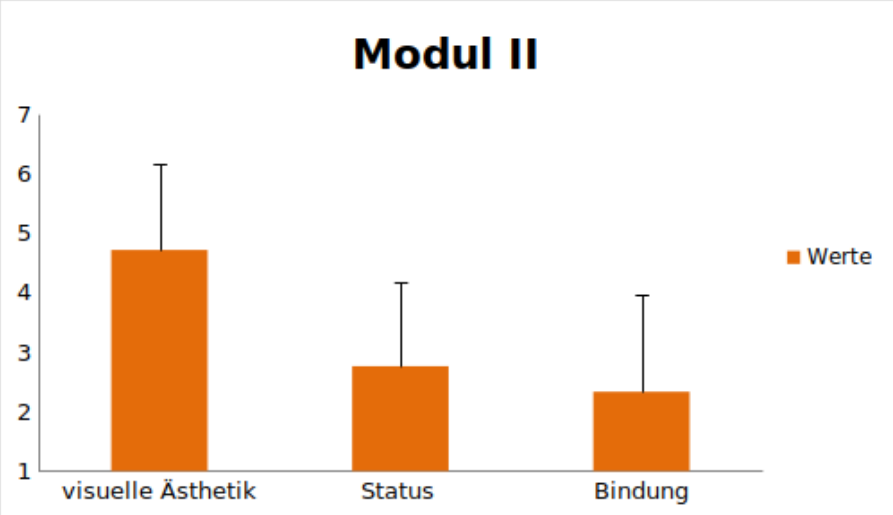

Abbildung 5.4: Die Mittelwerte der Antworten der Probanden im meCUE 2.0-Fragebogen. Der schwarze Strich gibt jeweils die Standardabweichung an.

5.7 DISKUSSION

Die Ergebnisse deuten, wenn man die Studie als vollständig gültig ansehen will (was man mit Blick auf die Threads to Validity nicht zwangsweise tun sollte), darauf hin, dass das Framework die Erwartungen nicht ganz erfüllt. Innerhalb der Studie hat das Framework die Bearbeitungszeit eher verlängert. Währenddessen konnte die Korrektheit durchaus gesteigert werden. Da man aber nicht in allen Use Cases immer eine exakte Zahl benötigt, sondern manchmal eine Schätzung genügt, sind kleinere Abweichungen teilweise hinnehmbar. Kleinere Verbesserungen in der Korrektheit sind somit nicht allzu relevant für uns.

Die Ergebnisse der Usability-Fragen sind meiner Meinung nach zufriedenstellend, dem SUS-Score zufolge dürfte das Framework einigermaßen benutzbar sein. Hier gibt es aber dennoch ein Verbesserungspotenzial. Der Design-Fragebogen fiel meiner Meinung nach ebenfalls zufriedenstellend aus. Das Design ist demzufolge etwa mittelmäßig gelungen, also nicht allzu schlecht. Auch hier gibt es ein Verbesserungspotenzial.

5.8 THREATS TO VALIDITY

Die Studie, die ich im Rahmen meiner Bachelorarbeit durchgeführt habe, weist einige deutliche Probleme auf, die die Gültigkeit in Frage stellen. Diese werde ich im Folgenden erläutern.

5.8.1 *Größe der Code Cities*

Die Code Cities für die vier gewerteten Aufgaben waren jeweils weniger als 100 Knoten groß. Man kann vermuten, dass diese Größe zu gering ist, um einen deutlichen Vorteil in der Bearbeitungszeit mit dem Framework zu sehen.

5.8.2 *Repräsentative Aufgaben*

Es kann sein, dass die gestellten Aufgaben nicht repräsentativ für typische Aufgaben in SEE sind. Dementsprechend hätten wir gezeigt, dass das implementierte Framework für die gegebenen Aufgaben nützlich ist, aber nicht, dass es für Aufgaben nützlich ist, die tatsächlich regelmäßig in SEE bearbeitet werden müssen.

5.8.3 *Fehler der Probanden*

ÜBUNGSAUFGABE FALSCH VERSTEHEN Bei der Pilotstudie hat ein Proband die Übungsaufgabe ohne ein Metrics Board bearbeitet, indem er die Blattknoten „von Hand“ gezählt hat. Somit konnte bei ihm nicht sichergestellt werden, dass er bereits ein Metrics Board erstellt hat, bevor er das erste Mal in einer gewerteten Aufgabe ein Board erstellt. Es kann also vorkommen, dass Probanden für die Aufgaben mit Metrics Board dennoch viel Zeit benötigen, weil sie sich die Anleitung zu den Metrics Boards nicht vorher durchgelesen haben. Das wollten wir eigentlich mit der Übungsaufgabe sicherstellen. Wenn die Probanden die Aufgabenstellung hier ignorieren, können sie trotzdem weiterkommen.

FALSCHES ANTWORTFELD Ein Proband hat davon berichtet, bei einer Aufgabe die Anzahl Kanten in das Feld für die Anzahl Knoten eingetragen zu haben. Dementsprechend ist hier die große Differenz zum korrekten Wert nicht darauf zurückzuführen, dass sich der Proband verzählt hat, sondern darauf, dass es einen Fehler im Ausfüllen des Fragebogens gab. Allerdings muss man ohnehin offen lassen, wie viele andere Probanden vielleicht auch beim Ausfüllen des Fragebogens Fehler gemacht haben, Aufgaben falsch verstanden haben etc.

NICHT MANUELL GEZÄHLT Ein anderer Proband hat angegeben, die Aufgaben immer mit den Boards bearbeitet zu haben, also nie manuell

die Knoten und Kanten gezählt zu haben. Hier hatte der Proband scheinbar die Aufgabenstellung nicht richtig gelesen oder verstanden. Dadurch könnte es bei diesem Probanden so aussehen, als hätte er die Aufgaben ohne Metrics Board dennoch mit gleicher Korrektheit und ähnlichem Zeitaufwand bearbeitet, wie die Aufgaben, die er mit Metrics Board bearbeiten sollte. Da ich dies nur zufällig, auf meine Nachfrage hin, im Gespräch mit diesem Probanden erfahren habe, wäre es auch denkbar, dass andere Probanden einen ähnlichen Fehler gemacht haben. Letztlich kann es somit sein, dass die Ergebnisse der Studie den Mehrwert der Metrics Boards deutlich geringer anzeigen, als er es tatsächlich ist.

5.8.4 *Mangelnde Übung*

Die Probanden hatten keine Erfahrung mit SEE und mit dem Framework. Somit liegt die Vermutung nahe, dass die Bearbeitungszeiten maßgeblich dadurch beeinflusst wurden, wie sicher die Probanden im Umgang mit der jeweiligen Technologie waren. Es wäre theoretisch wünschenswert gewesen, die Probanden hätten allesamt genug Erfahrung mit SEE und dem Framework gehabt. Somit hätte man präziser ermittelt, wie schnell man die Aufgaben an sich durch Abzählen beziehungsweise durch das Framework bearbeiten kann.

6 ZUSAMMENFASSUNG UND AUSBLICK

6.1 ZUSAMMENFASSUNG

Im Rahmen dieser Arbeit wurde ein Konzept entwickelt, um SEE um eine Komponente zu erweitern, mit der holistische Metriken angezeigt werden können. SEE wurde erfolgreich um eine solche Komponente erweitert, die bereit ist, eingesetzt zu werden. Am Ende wurde eine Nutzerstudie durchgeführt, bei der es vielleicht hilfreich gewesen wäre, auch einige SEE-Entwickler einzubeziehen. Bei den tatsächlichen Teilnehmern gab es zu viele Probleme mit der Bedienung von SEE.

Letztlich sind mir keine anderen Arbeiten auf Deutsch oder Englisch bekannt, die sich mit einer vergleichbaren Fragestellung befasst haben, wie diese Arbeit. Insofern ist diese Arbeit etwas relativ Neues.

6.2 AUSBLICK

Wie man an den Threads to Validity sehen kann, gibt es ernsthafte Gründe, die Gültigkeit der Ergebnisse der Studie in Frage zu stellen. Daher könnte man theoretisch eine erneute Evaluation zum Framework durchführen, bei der man versucht, diese Threads to Validity auszuschließen.

Außerdem könnte man das implementierte Framework anpassen oder erweitern. So wäre es zum Beispiel interessant, die Bedienung des Frameworks nicht hauptsächlich über ein Menü umzusetzen, sondern über Bedienelemente auf den Metrics Boards. Das könnte beispielsweise ein Knopf auf den Boards sein, über den man direkt ein neues Widget auf diesem Board hinzufügen kann.

Das Design hat von den Probanden eine akzeptable Bewertung erhalten, allerdings gibt es hier noch Verbesserungspotenzial. Man könnte also das Design der Metrics Boards und der Widgets überarbeiten.

6.3 KRITISCHE SELBSTREFLEXION

Zum Abschluss möchte ich noch eine Diskrepanz in meiner Vorgehensweise ansprechen. Bevor ich mit der Arbeit anfing, wurde im Exposé festgelegt, dass die Implementierung iterativ stattfinden soll. Dazu wäre es angemessen gewesen, regelmäßig Rücksprache mit meinem

Betreuer zu halten und den aktuellen Stand der Implementierung zu besprechen. Dies habe ich jedoch offenbar nicht häufig genug getan, was in folgendem Sachverhalt widergespiegelt wird: Gegen Ende der Implementierungsphase hatte ich ein Meeting mit meinem Betreuer. Er merkte an, dass die Bedienung des Frameworks teilweise aus dem Menü in die Spielwelt „verschoben“ werden könnte. Damit war gemeint, dass sich beispielsweise der Knopf zum Hinzufügen eines Widgets nicht im Menü befinden müsste, sondern auch direkt auf den Metrik-Anzeigetafeln sein könnte. Eine ähnliche Änderung hätte man für viele der Menü-Optionen durchführen können. Für einen so großen Umbau der Architektur war es jedoch für dieser Arbeit leider zu spät. Dennoch hat die Arbeit durchaus einen Mehrwert. Wenn auch die Bedienbarkeit des Frameworks vielleicht noch nicht optimal ist, wurde dennoch eine Lösung entwickelt, zu der sich in der Literatur bisher (soweit ich dies nach meiner Recherche beurteilen kann) kaum andere Ansätze finden.

A ANHANG

- `Bachelorarbeit.pdf` - Diese Arbeit als PDF-Datei.
- `Ergebnisse_Gruppe_A.xlsx` - Die Ergebnisse der Studie von Gruppe A in einer Excel-Datei, unbearbeitet.
- `Ergebnisse_Gruppe_B.xlsx` - Die Ergebnisse der Studie von Gruppe B in einer Excel-Datei, unbearbeitet.
- `Fragebogen_Gruppe_A.xlsx` - Die Excel-Datei für den Fragebogen für die Studie für Gruppe A.
- `Fragebogen_Gruppe_B.xlsx` - Die Excel-Datei für den Fragebogen für die Studie für Gruppe B.
- `SEE.zip` - Die Version von SEE für die Studie, mit den Code Cities.
- `Anleitung_Code_Cities.docx` - Eine Erklärung der Code Cities für die Probanden.
- `Anleitung_Metrics_Boards.docx` - Eine Anleitung zur Bedienung der Metrics Boards für die Probanden.
- `https://github.com/uni-bremen-agst/SEE/pull/512` - In diesem Pull Request ist fast der ganze Code enthalten, der für diese Arbeit geschrieben wurde. Das Repository ist allerdings privat. Um Zugriff zu erhalten, müsste beispielsweise der Erstgutachter dieser Arbeit, Rainer Koschke, kontaktiert werden.
- `intro_cities.mp4` - Das Video, das eine Erklärung der Code Cities in SEE enthält (für die Probanden der Studie).
- `intro_boards.mp4` - Das Video, das eine Anleitung für die Metrics Boards enthält (für die Probanden der Studie).

B

GLOSSAR

C# Das ist eine Programmiersprache, die unter anderem innerhalb der Unity-Engine für eigene Skripte verwendet werden kann. 9

Code City In der Code-City-Metapher werden Softwarekomponenten durch Gebäude in einer Stadt repräsentiert, wobei die Eigenschaften dieser Gebäude verschiedene Metriken der Software ausdrücken können — z. B. könnte die Höhe eines Gebäudes der Anzahl der Codezeilen entsprechen. iii, 1, 6, 7, 9–11, 14–17, 22, 23, 27, 31, 33

Convenience Sampling Convenience Sampling bedeutet, dass die Probanden aufgrund ihrer leichten Verfügbarkeit ausgewählt werden. 22

Framework Das Framework, das im Rahmen dieser Bachelorarbeit implementiert wurde, um Werte holistischer Metriken zu visualisieren. 2, 5–7, 9–11, 13–17, 19–21, 24–30, 35

SEE Das steht für **S**oftware **E**ngineering **E**xperience und ist ein Programm, mit dem Software in Form von Code Cities visualisiert wird. Dabei baut SEE auf der Unity-Engine auf. iii, 1, 5–7, 11, 14–17, 20–23, 27–29, 31

Szene Das ist in der Unity-Engine sozusagen **eine** Spielwelt. Die Szene von SEE, in der ich für meine Bachelorarbeit gearbeitet habe, besteht hauptsächlich aus einem Boden, dem Avatar des Spielers und einem Tisch, auf dem man eine Code City laden und betrachten kann. 6

UI Steht für User Interface. Das ist eine Oberfläche, über welche der Nutzer einer Software mit dieser interagiert. 6

Unity Unity ist eine Videospiel-Engine, also eine Software, die als Rahmenwerk bei der Spieleentwicklung genutzt werden kann. 6, 9

C

ABBILDUNGSVERZEICHNIS

LITERATURVERZEICHNIS

[Bangor u. a. 2009] Bangor, Aaron ; Kortum, Philip ; Miller, James: Determining what individual SUS scores mean: Adding an adjective rating scale. In: *Journal of usability studies* 4 (2009), Nr. 3, S. 114–123

[Brooke 1996] Brooke, J: *Usability evaluation in industry, Kap. SUS: a "quick and dirty" usability scale*. 1996

[Davis 1985] Davis, Fred D.: *A technology acceptance model for empirically testing new end-user information systems: Theory and results*, Massachusetts Institute of Technology, Dissertation, 1985

[Figl 2009] Figl, Kathrin: ISONORM 9241/10 und Isometrics: Usability-Fragebögen im Vergleich. In: *Mensch & Computer* Bd. 9, 2009, S. 143–152

[Galperin 2021] Galperin, Falko: *Visualisierung von Code-Smells in Code-Cities*, Universität Bremen, Bachelorarbeit, 2021

[Minge und Riedel 2013] Minge, Michael ; Riedel, Laura: meCUE-Ein modularer Fragebogen zur Erfassung des Nutzungserlebens. In: *Mensch & Computer*, 2013, S. 89–98

[Rummel und Ruegenhagen 2015] Rummel, Bernard ; Ruegenhagen, Eva: *System Usability Scale – jetzt auch auf Deutsch.* 2015. – URL https://web.archive.org/web/20200607035254/https://experience.sap.com/skillup/system-usability-scale-jetzt-auch-auf-deutsch/. – Zugriff am 27.12.2022