Universität Bremen
Bachelorarbeit
Fachbereich 3 - Mathematik und Informatik

Laufzeitkonfiguration von SEE-Cities

Vorgelegt von:

Ruben Smidt
Stedinger Str. 24
28203 Bremen

Email: rsmidt@uni-bremen.de
Matrikelnummer: 4433503
Studiengang: Wirtschaftsinformatik

Erstgutachter:in: Prof. Dr. Rainer Koschke
Zweitgutachter:in: Dr.-Ing Tanja Döring

Abgabe: 14. Juni 2021

Zusammenfassung

In dieser Arbeit wird vorgestellt, wie die Implementierung einer Benutzerschnittstelle die Laufzeitkonfiguration von Software-Engineering-Experience-Städten erlaubt. Dabei wird die Bedienbarkeit der erarbeiteten Schnittstelle mittels einer Nutzerstudie untersucht. Ziel ist es herauszustellen, ob die wahrgenommene Benutzerfreundlichkeit durch die Proband:innen zwischen zwei Eingabemedien variiert. Es wird gezeigt, dass die Benutzerfreundlichkeit als befriedigend sowohl im Einsatz am Desktop als auch in Virtual Reality wahrgenommen wird und es keine signifkanten Abweichungen im Vergleich zwischen den Eingabemedien gibt. Im Anschluss wird ein Ausblick gegeben, der sich primär aus dem Feedback der Proband:innen zusammensetzt.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Bremen, den _____

_____
(Ruben Smidt)

Vorwort

In Zeiten einer globalen Pandemie und den damit einhergehenden Einschränkungen möchte ich mich außerordentlichst bei allen Menschen bedanken, die mich in verschiedensten Formen bei der Erstellung dieser Arbeit unterstützt haben. In allerster Linie möchte ich mich bei Prof. Dr. Rainer Koschke bedanken, der stets Hilfestellungen angeboten hat und diese Arbeit mit viel Geduld betreut hat.

Gerade unter Kontaktbeschränkungen ist es schwierig, eine Nutzerstudie in Präsenz durchzuführen. Daher gilt ein besonderer Dank allen Teilnehmerinnen und Teilnehmern, die durch ihre Teilnahme einen großen Teil zu dieser Arbeit beigetragen haben.

Ein persönlicher Dank gilt meinem geschätzten Kommilitonen und Freund Hannes Masuch, der mich allzeit bei Fragen zu der Arbeit mit Unity und Software Engineering Experience (SEE) unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

API Application Programming Interface.

GUI Graphical User Interface.

HCI Human-Computer Interaction.

IDE Integrated Development Environment.

MR Mixed Reality.

SEE Software Engineering Experience.

SMEQ Subjective Mental Effort Question.

SUMI Software Usability Measurement Inventory.

SUS System Usability Scale.

UX User Experience.

VR Virtual Reality.

Glossar

Game Engine Das Kernstück eines Videospiels, das das Ausführen des Spiels überhaupt erst erlaubt..

Reverse Engineering Das Schaffen von Informationen und Fakten über ein bereits fertiges System, durch Analyse eben dieses Systems.

1. Einleitung

Während durch immer größer werdende Software auch unweigerlich die Komplexität steigt, steigt damit auch der Bedarf an Methoden, diese Komplexität verständlich und wartbar zu machen (Yu & Zhou, 2010). Ein Großteil des Wartungs- und Korrekturaufwands wird bereits für das Verständnis von Software aufgewandt - Annahmen gehen von 47% bis 62% der Wartungs- und Korrekturzeit lediglich für das Erlangen von Verständnis über die Software aus (Panas et al., 2003).

Eine gut dargestellte Analyse der Software kann die Komplexität eben dieser Software vereinfachen, was zu einer besseren Verständlichkeit führen kann. Yu und Zhou (2010) schlagen zum Lösen dieses Problems die Visualisierung von Reverse-Enginnered-Softwaremetriken mittels einer Metapher vor, die allen Stakeholdern ein schnelles Verständnis von der vor ihnen liegenden Analyse gibt. Dabei ist der dreidimensionale Charakter der Metapher entscheidend.

Wettel und Lanza (2007) greifen diesen Ansatz auf und entwickeln eine Städte-Metapher, die gerade im Bezug auf objektorientiert entwickelte Software die Komplexität einer Software verständlich für diverses Publikum visualisieren soll. Diese Städte-Metapher ist eine der Visualisierungen, die das von Prof. Dr. Rainer Koscke ins Leben gerufene Projekt SEE unterstützt.

1.1. Problemstellung

Eine SEE-Stadt, mit ihren Häuser, Straßen und Distrikten, ist mit einer Vielzahl an Optionen konfigurierbar. Dabei ist es dem/der Anwender/in überlassen, zu entscheiden, welche konkreten Metriken beispielsweise die Dimensionen von Häusern bestimmen. So kann zum Beispiel die Höhe eines Hauses anhand der Zeilen Quelltext eines Moduls der Software bestimmt werden. Hierfür können aber auch andere Metriken oder gar fixe Werte benutzt werden. Diese Konfiguration wird vorgenommen, bevor eine Stadt gezeichnet wird, mittels des verwendeten Integrated Development Environment (IDE) der Game Engine Unity, bevor die Software (das Spiel) gestartet und bedient wird. Folglich ist IDE aktuell zwingend notwendig, um die Software nach eigenen Wünschen zu konfigurieren und damit überhaupt erst richtig benutzbar zu machen.

1.2. Ziel der Arbeit und Vorgehensweise

Damit SEE also auch paketiert und ausgeliefert bedient werden kann, muss eine Schnittstelle zur Konfiguration während der Laufzeit im Spiel bereitgestellt werden. Eine Solche Bedienmöglichkeit ist im Zuge dieser Arbeit erörtert, implementiert und evaluiert worden. Zunächst wurde mithilfe von Konzeptillustrationen einer möglichen Benutzeroberfläche geprüft, ob sich alle, zum Zeitpunkt der Erhebung verfügbaren, Konfigurationsoptionen einer Stadt abbilden und zur Laufzeit bedienen lassen würden.

Das daraus entstandene Konzept wurde genutzt, um eine Implementierung in SEE vorzunehmen und die Schnittstelle sowohl für den Desktop- als auch für den Virtual-Reality-Gebrauch zugänglich zu machen. Das Ergebnis wurde dann in den regulären Ablauf des Spiels integriert.

Um die Bedienbarkeit der Benutzeroberfläche zu bewerten, wurde eine Nutzerstudie durchgeführt. Dabei wurde untersucht, wie die Bedienbarkeit der selben Komponente variiert von dem Einsatz am Desktop und in Virtual Reality (VR).

1.3. Aufbau dieser Arbeit

Die Einführung in das Problem und das Ziel dieser Arbeit erfolgt im ersten Kapitel. Im zweiten Kapitel wird ein Grundverständnis über die in dieser Arbeit relevanten Terminologien und Technologien vermittelt. Dazu zählt unter anderem die verwendete Game Engine und die Software zur Erstellung von Konzeptillustrationen. Inhalt des dritten Kapitels ist das, der Implementierung zugrunde liegende, Konzept der Benutzeroberfläche. In Kapitel 4 wid die Implementierung und dazugehörige Details erläutert und besprochen. Die Ergebnisse der Evaluation und die Auswertung an sich werden in Kapitel 5 diskutiert. Im letzten Kapitel wird ein Ausblick über weitere Entwicklungsmöglichkeiten geboten.

2. Grundlagen

Ein Artefakt dieser Arbeit ist eine graphische Benutzeroberfläche, die als Teil von SEE ausgeliefert wird. Die dafür notwendigen Berücksichtigungen und Entscheidungen werden im Folgenden erläutert, sowie weitere Themen, die relevant für das Verständnis, nicht nur des Benutzeroberflächen-Artefakts sind, sondern auch für die Evaluation.

2.1. Software Engineering Experience (SEE)

Wie in der Einleitung bereits erwähnt, ist das SEE-Projekt ein Versuch, die Analyse von Software zu vereinfachen, mittels Visualisierung im dreidimensionalen Raum. Dabei geht es aber auch primär um die Visualisierung. Das Beschaffen der Metriken zur Analyse muss andererorts durchgeführt werden.

Während die Betrachtung eines bestimmten Zeitpunkts im Leben einer Software möglich ist, so lassen sich mit SEE aber auch andere Anwendungsszenarien realisieren: ein Abgleich zwischen der Architektur (dem angedachten Idealzustand der Software) und der Implementierung (dem tatsc̈hlichen Resultat der Entwicklung) vereinfacht es den Entwickler:innen die Komplexität ihrer Software greifbar zu machen und aufzuzeigen, wo eine Divergenz zwischen Architektur und Implementierung zu Problemen führen könnte.

Durch die Visualisierung im dreidimensionalen Raum mittels menschennahen Metaphern wie beispielsweise der *Code City* ist es aber auch anderen Stakeholdern als den Entwickler:innen möglich, ein Verständnis über den Zustand der Anwendung zu erhalten (Koschke, 2020).

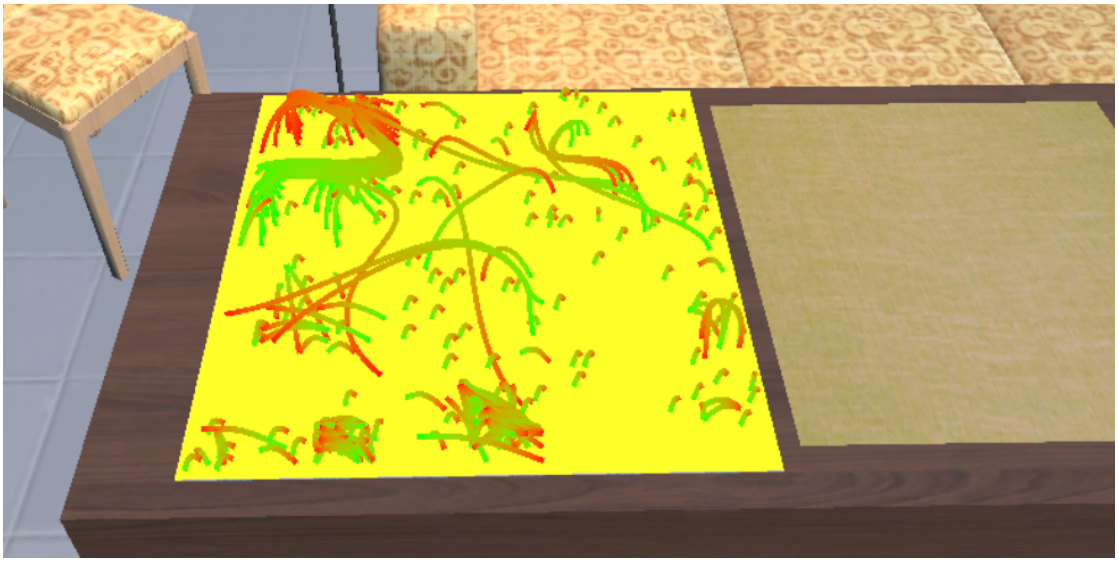

Abbildung 1: *Beispiel einer Visualisierung mit SEE*

2.2. Unity

Unity ist die Game Engine, auf der SEE fußt. Unity kann zum einen plattformübergreifend zur Entwicklung genutzt werden, die mit Unity entwickelte Software lässt sich aber auch plattformübergreifend ausführen - vorausgesetzt systemspezifische Abhängigkeiten werden beachtet. Ein Beispiel hierfür ist der Einsatz der Microsoft Windows-Speech-Recognition-API, die den Einsatz der betriebssystemeigenen Spracherkennung in Unity ermöglicht. Diese Arbeit greift auf die Windows-Speech-Recognition-API zurück, was den Einsatz der Spracherkennung auf anderen Systemen nicht ermöglicht. Unity bietet zudem auch Unterstützung für VR und Mixed Reality (MR), die auch Zielgruppe von SEE sind.

2.3. Graphical User Interfaces

Ein Graphical User Interface (GUI) ist die Schnittstelle, die in der meisten, modernen Software zu finden ist (Stone, 2005). Dabei besteht eine GUI aus diversen Interaktionselementen, wie beispielsweise Formularfeldern zur Datenerfassung, aber auch direkten Aktionselementen, unter anderem Knöpfen und Auswahlfeldern. Dabei obliegt die Strukturierung freier Entscheidung.

Bei der Entwicklung einer GUI kommt es nicht nur auf das reine Illustrieren eines Eingabeelements (z. B. eines Texteingabefelds) an, sondern auch auf das Entwerfen einer User Experience (UX). Der Begriff UX ist dabei nach Erkenntnissen von Hassenzahl und Tractinsky (2006) in akademischen Kreisen nicht final definiert, beschreibt aber im Technologiekontext eine Technologie, die mehr als nur einen praktischen Zweck erfüllt, sondern diverse Aspekte berücksichtigt. Solche Aspekte können laut den Autor:innen z. B. ästhetischer Natur sein, d. h. es wird nicht nur erwartet, dass das Produkt nutzbar, sondern auch noch visuell ansprechend ist. Die UX berücksichtigt aber auch die Emotionen, Gemütszustände der Nutzer:innen und hedonische Aspekte wie den Wunsch nach persönlichem Wachstum. Während es schon immer ein Ziel der Human-Computer Interaction (HCI) war, negative Emotionen der Nutzenden zu vermeiden, so versucht die UX gar, positive Emotionen hervorzurufen.

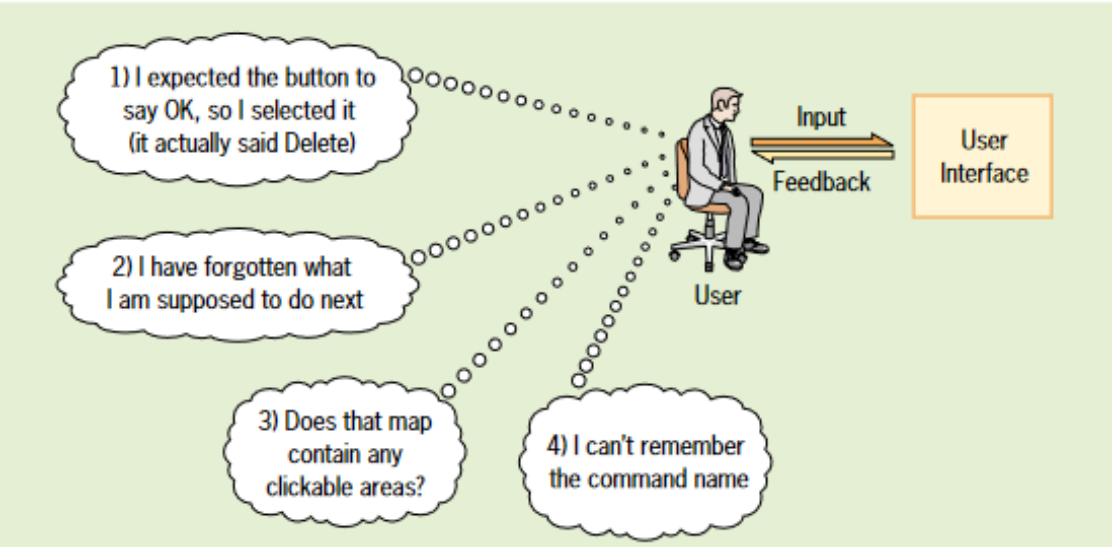

Abbildung 2: *Missachten der grundlegenden Prinzipien führt zu Verwirrung.*
Quelle: Stone, 2005

3. Entwurf

Dieses Kapitel stellt die getroffenen Entscheidungen und Ideen vor, die der Implementierung des zu untersuchenden Forschungsgegenstands zugrunde liegen. Dabei wird unterschieden zwischen den allgemeingültigen Anforderungen an eine GUI und den spezifischen Anforderungen an eine Benutzerschnittstelle zur Konfiguration einer SEE-Stadt.

3.1. Anforderungen an eine GUI

Wie in Kapitel 2.3 erwähnt muss zum einen gewährleistet sein, dass die funktionalen Anforderungen (der praktische Zweck) der GUI erfüllt ist, aber auch die UX muss als Teil des Schnittstellenentwurfs berücksichtigt werden.

Stone (2005) bedient sich grundlegender Annahmen, die in der Regel auf menschliche Psychologie zurückgeführt werden. Diese Annahmen helfen, Nutzerverhalten zu charakterisieren und dadurch Prinzipien aufzustellen, die beim Entwurf und der Implementierung von Benutzerschnittstellen berücksichtigt werden sollen. Die Autor:innen zählen auf:

Users See What They Expect to See
Durch bereits gesammeltes Wissen und Erfahrungen erwartet der Nutzende in bestimmten Situation bestimmte Ergebnisse. Das impliziert, dass unerwartete Situationen für Nutzende schwierig zu verarbeiten sein können, wenn Muster, in denen der Nutzende überlicherweise denkt, nicht zutreffen.

Als Beispiel wird der in fast jedem Betriebssystem enthaltene Taschenrechner aufgeführt. Durch Erfahrungen mit physischen Taschenrechnern hat der Nutzende ein bestimmtes Bild vor Augen, wie ein Taschenrechner zu bedienen ist. Wird nun versucht eine GUI für ein System zu entwickeln, das das Rechnen am Computer ermöglicht, liegt es Nahe, auf die Metapher des Taschenrechners zurückzugreifen. Die Nutzenden müssen nicht lange überlegen, um zu verstehen, was sie dort sehen, sondern können ihr vorheriges Wissen direkt nutzen, um das System zu bedienen. Dies wird von den Autor:innen als *principle of exploiting prior knowledge* bezeichnet. Carroll et al. (1988) zeigen ebenfalls, dass ein auf Metaphern beruhender Schnittstellenentwurf die Benutzerfreundlichkeit verbessert.

Es setzt aber auch voraus, dass eine Grundkonsistenz im Layout und der Gestaltung der GUI vorhanden ist. Bestimmte Farben werden mit bestimmten Informationen und Kontexten verknüpft. Rot steht in der Regel für Gefahr, während Grün einen Erfolgsfall andeutet. Mit diesen Erwartungshaltungen der Nutzenden sollte nicht gebrochen werden - zusammengefasst durch die Autor:innen als *principle of consistency*.

Users Have Difficulty Focusing on More Than One Activity at a Time
Während es Nutzenden möglich ist, ihre Aufmerksamkeit zwischen mehreren Aktiviäten zu teilen, so kann dies von Nutzendem zu Nutzendem variieren. Daher nennen die Autor:innen zwei Prinzipien, denen ein Schnittstellenentwurf folgen soll. Dem *principle of perceptual organization* folgend, sollen Elemente, die kontextuell zusammengehörig sind, auch visuell beziehungsweise strukturell gruppiert werden. So können sich die Nutzenden auf einen Kontext konzentrieren. Wenn eine Information oder eine Interaktion wichtig für den Nutzenenden ist, dann sollte diese auch visuell hervorgehoben werden - ganz dem zweiten Prinzip, dem *principle of importance* folgend.

It Is Easier to Perceive a Structured Layout
Den Autor:innen zufolge ist es für Nutzende einfach, Dinge wahrzunehmen, die strukturell in Gruppen arrangiert sind. Nach der Gestalt Psychologie liegen dieser Wahrnehmung einige Gesetze zugrunde (Köhler, 1970), die ebenfalls ausschlaggebend für den Entwurf einer GUI sind. Nach dem *law of proximity* werden Elemente, die dicht zusammen dargestellt werden, eher als Gruppe und nicht als einzelne Elemente wahrgenommen. Das *law of similarity* wiederum hat als Kernaussage,

dass Elemente derselben Farbe oder Form erscheinen, als würden sie zusammengehören.

It Is Easier to Recognize Something Than to Recall it
Die Autor:innen behaupten, dass es für Menschen einfacher ist, Informationen zu erkennen (z. B. abzulesen), als Informationen aus dem Gedächtnis abzurufen. Daher sollte im Entwurf einer Benutzerschnittstelle das erklärte Ziel sein, Informationen, die der Nutzende braucht, auch zu präsentieren. Dabei ist die Darstellung dieser Informationen wieder frei wählbar. Als Beispiel werden Icons gennant. Diesen Umstand postulieren die Autoren als *principle of recognition*, d. h. wenn möglich sollte man Nutzenden Informationen direkt präsentieren und nicht erwarten, dass diese Informationen eventuell im Gedächtnis des Nutzenden vorhanden sind.

Abschließend lassen sich die Prinzipien, die nach Stone (2005) ein gutes UI-Design ausmachen, wie folgt aufzählen:

- P1: *principle of exploiting prior knowledge*
- P2: *principle of consistency*
- P3: *principle of perceptual organization*
- P4: *principle of importance*
- P5: *principle of recognition*

Die Erkenntnisse von van Welie et al. (2001) unterstreichen die Bedeutsamkeit von Pattern in dem Design von Benutzerschnittstellen. Laut den Autor:innen reichen einfache Regeln für das Design einer GUI nicht aus. Unter Regeln werden Anweisungen verstanden, die beispielsweise das Platzieren der Eingabelemente an bestimmten Stellen vorsehen. Als Beispiel wird eine Regel wie "Das Label eines Eingabelements muss immer linksbündig platziert werden." aufgeführt. Diese Regel stellt eine klare Anweisung dar, beschreibt aber nicht, warum dies passieren muss beziehungsweise welchen Mehrwert dies für die Benutzererfahrung hat.

In der folgenden Betrachtung der Konzeptzeichnungen, werden diese Prinzipien aufgegriffen und dienen zur Rechtfertigung des gewählten Designs.

3.2. Konzeptzeichnungen

Wie im u den GUIs beschrieben, besteht eine GUI aus vielen verschiedenen Eingabeelementen. Diese Eingabeelemente abstrahieren den Zugriff auf einen bestimmten Datenpunkt. In einem Login-Formular dient ein Texteingabefeld zur

E-Mail-Erfassung, während in einem Reisebuchportal ein *date picker* (Datumsauswahlfeld) genutzt wird, um die An- und Abreisedaten auszuwählen. Um das erste Prinzip korrekt anzuwenden, sollte also auch für die Implementierung der GUI dieser Arbeit auf vertraute Elemente zuŕuckgegriffen werden:

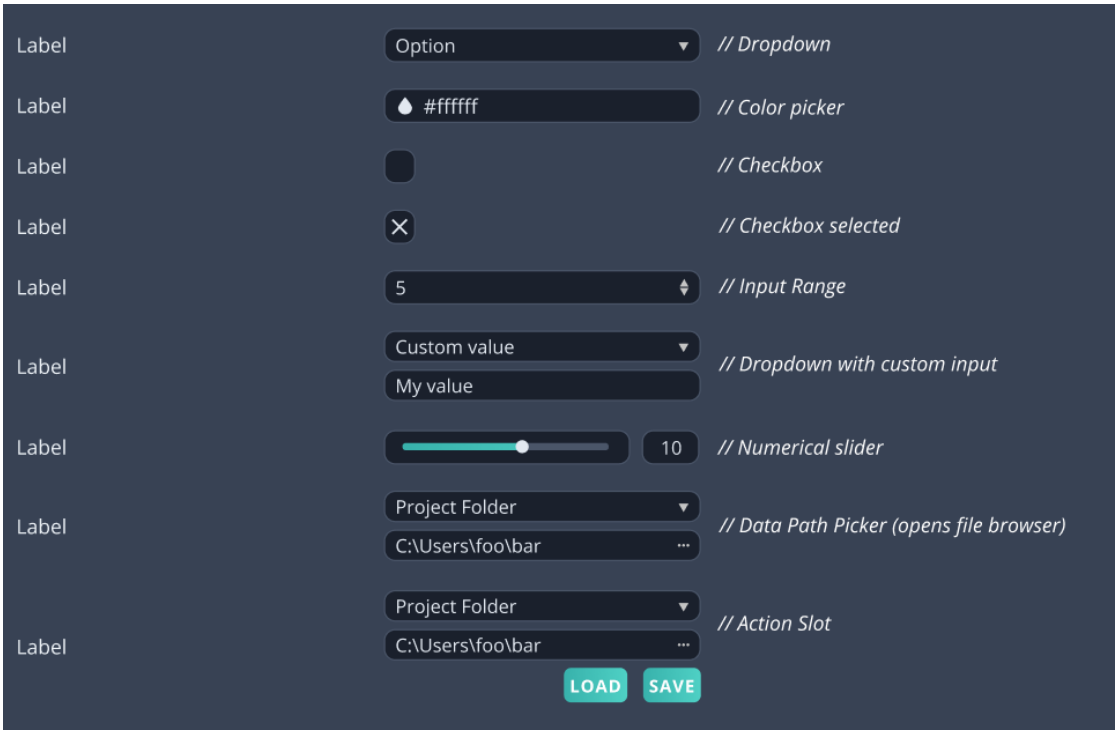

Abbildung 3: *Übersicht über alle angedachten Eingabeelemente*

Im Folgenden wird die Zusammensetzung einer Menu-Seite am Beispiel der *Nodes and node Layout* Seite gezeigt. Diese Seite setzt sich aus sieben Eingabeelementen zusammen, die jeweils einen konkreten Datenpunkt darstellen und manipulieren.

Leaf nodes Es gibt eine (aktuell) abzählbare und finite Anzahl an Möglichkeiten, die *leaf nodes* der Code-City zu visualisieren. Eine freie Eingabe mittels Texteingabefeld ist also fehlplatziert und würde den Nutzenden nur verwirren. Durch Aufrufen des *dropdown* (Auswahlfelds) erhält der Nutzende sofort einen Überblick über die Auswahlmöglichkeiten. Die gleichen Überlegungen gelten auch für *node layout*.

Layout file Dieses Datum ist ein Beispiel für die Eingabe von arbiträren Binär- oder Nichtbinärdateien (Texte, Bilder usw.) mittels *file picker* (Dateiauswahl).

Eine Besonderheit ist hier die Zusammensetzung der Eingabegruppe aus zwei Eingabefeldern, einem *dropdown* und einem *file picker*. Mittels des *dropdown* kann der Suchort der Datei im Vorhinein eingeschränkt werden auf beispielsweise den Projektordner. Dem dritten Prinzip folgend wird das Label der beiden Inputs vertikal in Bezug auf die beiden Eingabefelder platziert, um eine visuelle Gruppe zu erzeugen.

Z-score scaling Um einen binären Zustand (ja oder nein) abzubilden, wird auf bereits bekanntes Wissen zurückgegriffen. Eine *checkbox* (Häckenfeld) visualisiert mittels Kreuz, ob eine Auswahl zutrifft oder nicht. Das ist den Nutzenden beispielsweise durch Schultests bereits geläufig. *Show erosions* folgt dem gleichen Gedanken.

Max. width of erosion icon Die Eingabe von Zahlen kann sehr stark variieren, je nach Anwendungsszenario und Anforderungen. Wird beispielsweise nur ein bestimmter Rahmen an auswählbaren Zahlen zugelassen oder ist der Schritt zwischen zwei Zahlen klar definiert, so wird oftmals auf den *stepper* (schritteingeschränkte Zahlenauswahl). Das Erhöhen oder das Verringern der Zahl um den fest vorgegebenen Schritt erfolgt meist über Symbole direkt im Eingabefeld. Da diese jedoch in der Regel nur schwierig in VR zu bedienen sind, wurde stattdessen der *range slider* (Schieberegler) in Betracht gezogen. Der Nachteil des *range slider* ist, dass nur ein fester Zahlenbereich ausgewählt werden kann, in diesem aber fließend Zahlen ausgewählt werden können

4. Implementierung

Inhalt dieses Kapitels ist das Vorstellen und Beschreiben der Implementierung. Dabei werden etwaige Problemstellungen und die daraus resultierten Abweichungen zum Konzept erläutert.

Die Benutzerschnittstelle wurde unter Einhaltung einiger Prämissen entwickelt. Eine dieser Prämissen ist der Einsatz einer externen Bibliothek *Modern UI*[1], die diverse Eingabeelemente bereitstellt, die zum Teil in Kapitel 3.2 vorgestellt wurden. Dies können Klickelemente (Buttons), aber auch Texteingabefelder sein. Der Einsatz einer solchen Abhängigkeit vereinfacht das Komponieren einer Benutzerschnittstelle dahingehend, dass die Elemente sich in ihrem initialen Design ähneln. Ein Nachteil, der durch den Einsatz solcher externen Komponenten entsteht, ist die eingeschränkte Erweiterbarkeit der Interaktionselemente. Die Bibliothek bietet nicht genug Komponenten, um alle geforderten Interaktionen abzubilden, weshalb teilweise Eingriffe in den Quelltext der Bibliothek vorgenommen werden mussten, um z. B. die angezeigten Dezimalstellen eines Labels zu vergrößern.

Des Weiteren wurde die Bibliothek *Curved UI*[2] vorgeschlagen, um die Interaktion mit der Benutzerschnittstelle in VR zu verbessern. Um das zu Erreichen, wird die gesamte Benutzerschnittstelle nach außen gewölbt, was es dem Nutzer oder der Nutzerin erlaubt, sich im Mittelpunkt des Zylinders aufzuhalten und lediglich durch Drehen des Kopfes alle Elemente zu erreichen. Ein Problem beim Einsatz dieser Bibliothek, das später noch genauer erläutert wird, ist die Inkompatibilität mit anderen Bibliotheken, z. B. *Modern UI*.

4.1. Kontrollierte und unkontrollierte Eingabeelemente

Die Benutzerschnittstelle soll die Konfiguration einer sogenannten SEE-City-Instanz erlauben. Dieses Objekt definiert die Attribute, die notwendig sind, um eine Code City darzustellen. In den meisten Fällen sind diese Werte skalar, d. h. es sind elementare Datentypen wie Gleitkommazahlen, Ganzzahlen, Zeichenketten und Wahrheitswerte. Es werden aber auch komplexere Datentypen verwendet, was für das Design einer einheitlichen Abstraktion der Eingabelemente auf technischer Ebene relevant ist.

Für das Manipulieren eines dieser Werte, ungeachtet der Komplexität, sind für diese Arbeit unter anderem zwei Methoden angewandt worden, die im folgenden

[1] https://assetstore.unity.com/packages/tools/gui/modern-ui-pack-150824
[2] https://assetstore.unity.com/packages/tools/gui/curved-ui-vr-ready-solution-to-bend-warp-your-canvas-53258

näher vorgestellt werden. Die Begriffe der kontrollierten und unkontrollierten Eingabeelemente so wie das Design der Elemente sind von Konzepten aus der Web-Entwicklung entliehen. Das von Facebook entwickelte Framework zur einfachen Zusammensetzung von Benutzerschnittstelle *React* ist hierbei ein prominentes Beispiel für den Einsatz der angesprochenen Konzepte[3].

Unkontrollierte Eingabeelemente

Ein unkontrolliertes Eingabelement verwaltet seinen Eingabewert selbst und Änderungen an den Daten von außerhalb der Instanz werden ignoriert. Wenn das Eingabeelement auf der Benutzerschnittstelle dargestellt werden soll, wird zunächst ein Startwert für dieses Element vorgegeben. Ändert sich nun der Wert durch Nutzereingaben, so wird die Änderung zunächst in dem Element selbst hinterlegt und nur gegebenfalls nach außen kommuniziert.

Im Kontext von SEE würde das bedeuten, dass wenn die zugrunde liegende SEE-City-Instanz anderweitig und nicht durch die Benutzerschnittstelle geändert werden würde, das Eingabeelement diese Änderung ignorieren würde.

Kontrollierte Eingabeelemente

Ein Eingabeelement gilt dann als kontrolliert, wenn es zwei Anforderungen erfüllt. Zum einen muss das Element immer den Eingabewert benutzen (sprich anzeigen), den das Element von außerhalb der Instanz vorgegeben bekommt. Eingaben der Nutzenden können also nur in dem Eingabeelement reflektiert werden, wenn die Eingabe direkt nach außen kommuniziert wird. Diese erzwungene Kommunikation stellt die zweite Anforderung dar.

Dieses Vorgehen hat den Vorteil, dass es nur eine *single source of truth* für die Daten einer SEE-City-Instanz gibt und alle Benutzerschnittstellen, die diese Instanz manipulieren, stets synchronisiert sind. Daher wurde versucht, alle Eingabefelder nach diesem Designprinzip zu entwickeln.

Formal lassen sich die Anforderungen als folgendes UML-Diagramm abbilden:

[3] https://reactjs.org/docs/uncontrolled-components.html Zugriff: 26.05.2021

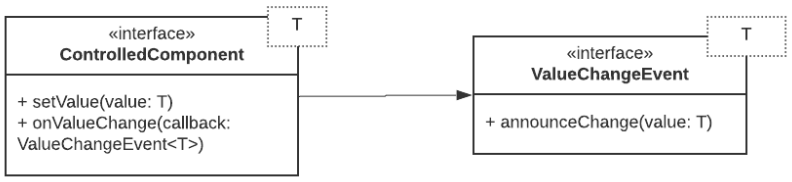

Abbildung 4: *Der Vertrag, den eine kontrollierte Komponente erfüllen muss.*

Je nach technischen Möglichkeiten der Implementierungssprache kann das gezeigte Interface variieren.

Der idealisierte und vereinfachte Ablauf der Interaktion mit einem kontrollierten Eingabeelement (hier auch als Komponente bezeichnet), wird im folgenden Sequenzdiagram dargestellt.

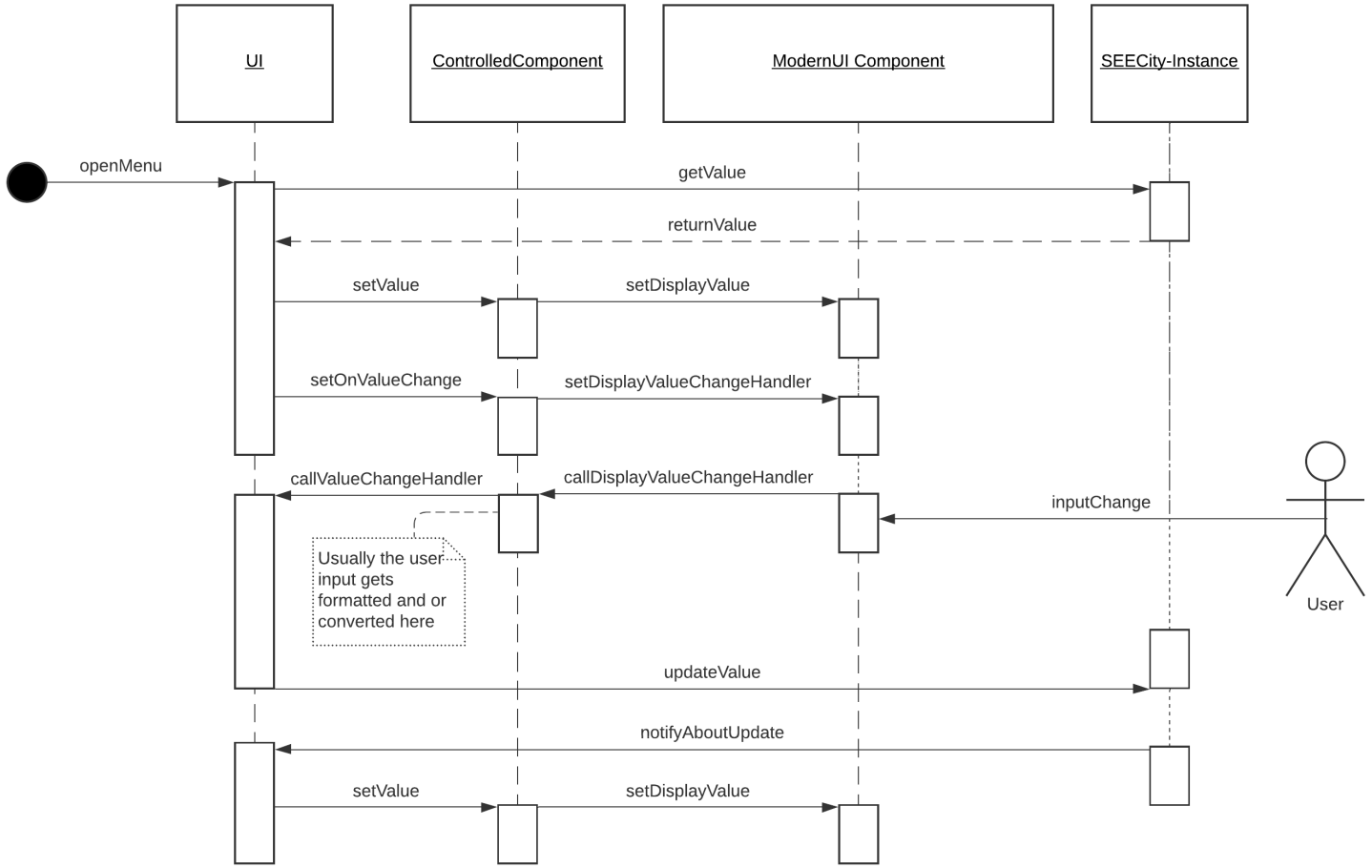

Abbildung 5: *Der idealisierte Ablauf einer Interaktion mit einer kontrollierten Komponente.*

4.2. Das Menu

Alle bereits vorgestellen Funktionen der Benutzerschnittstelle sind in einem Menu, einem Unity-Objekt gebündelt. Dem liegt ein Unity-*Canvas* zugrunde, der neben der visuellen Abtrennung auch eine Gruppierung für die einzelnen Eingabeelemente vornimmt.

Das Menu gliedert sich in drei grundlegende Unterkomponenten: die Seitenleiste, die angezeigte Menuseite und eine dynamisch eingeblendete Aktionsleiste, die visuell abgekoppelt von dem Rest des Menus dargestellt wird.

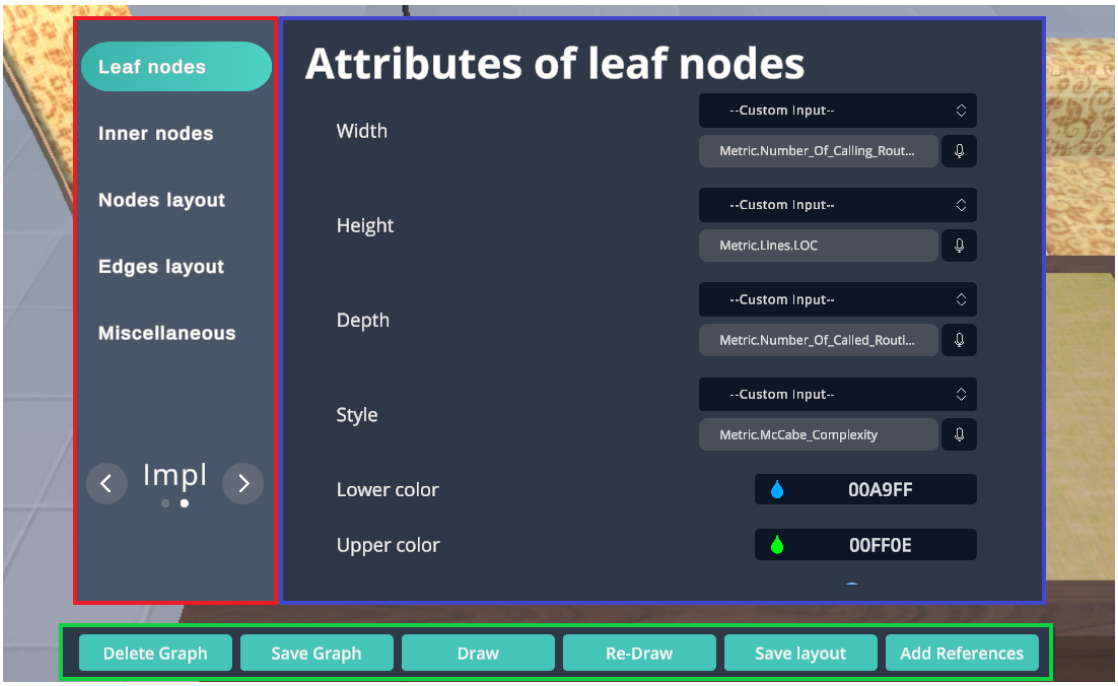

Abbildung 6: *Die Aufteilung des Menus. Rot: Seitenleiste, Blau: Aktuelle Seite, Grün: Aktionsleiste*

Die Seitenleiste
Da die vielfältigen Konfigurationsmöglichkeiten einer SEE-City in ihrer Anzahl zu groß für lediglich eine einzelne, gescrollte Seite sind, wurden die Konfigurationsmöglichkeiten semantisch gruppiert. Die Seitenleiste bietet hierfür fünf Buttons.

Zu dem gibt es eine Auswahl, welche Instanz der SEE-City mit dem Menu konfiguriert werden soll. Wie eingangs erwähnt, gibt es eine SEE-City zur Darstellung

der tatsächlichen Implementierung und eine SEE-City für die angedachte Architektur. Daher rühren auch die Abkürzungen *Impl* für Implementierung und *Arch* für Architektur.

Als letztes befindet sich ein Button in der Leiste, um die gewünschte Konfiguration zu laden und damit weitere Interaktionsmöglichkeiten in Form der Aktionsleiste darzustellen.

Die aktuelle Seite
Der Großteil des Menus ist für die eigentlichen Eingabeelemente vorgesehen. Daher ist hier auch nur eine Überschrift vorgesehen, der Rest sind Eingabeelemente in einem zweispaltigen Layout, bestehend aus einem Label und dem Eingabeelement. Auf manchen Seiten kann die Zahl der Eingabeelemente so groß werden, dass gescrollt werden muss. Eine Trennung der Elemente würde mit der semantischen Zusammengehörigkeit brechen.

Während es möglich ist, die Seite komplett per Unity-Editor aufzubauen, d. h. per graphischem Interface die einzelnen Eingabeelemente manuell zu platzieren, so sollte es bevorzugt möglich sein, das Menu programmtisch mit Eingabeelementen zu befüllen.

Die Aktionsleiste Der letzte Bestandteil ist des Menus ist die Aktionsleiste, die nach Laden des Graphens angezeigt wird. Die Interkationsmöglichkeiten sind ausschließlich als Buttons modelliert und orientieren sich an den bereits vorhandenen Graphoperationen: Löschen, Speichern, Zeichnen usw.

4.3. Erweiterbarkeit

Die möglichen Parameter, die den Aufbau, das Aussehen und das Verhalten einer Code-City beeinflussen, lassen sich stetig verändern, erweitern oder auch entfernen. Das hat zur Konsequenz, dass auch alle Stellen, die die Möglichkeit bieten, diese Parameter anzupassen, ebenfalls bei der Änderung berücksichtig werden müssen. Wenn ein neuer Parameter ergänzt wird (z. B. die Lautstärke von Soundeffekten), muss auch die Benutzerschnittstelle dahingehend angepasst werden. Daher bietet die hier implementierte Benutzerschnittstelle diverse Ansatzpunkte, um auch in Zukunft skalierfähig zu sein. Weitere Code-City-Instanzen können beispielsweise in die bereits vorgestellte Auswahl in der Seitenleiste integriert werden. Neue Parameter für bestehende Seiten können an einen beliebigen Platz der Seite platziert werden, ohne dass technische Limitierungen dies verhindern. Die Reihenfolge ist ebenfalls auswechselbar. Um das zu erreichen, werden die einzelnen Seiten des Menus dynamisch aufgebaut. Mittels Builder-Pattern werden einzelne Eingabeelemente (inklusive des dazugehörigen Labels) instanziiert und der Seite angehängt (Gamma,

Erich and Helm, Richard and Johnson, Ralph and Vlissides, John M., 1995). Der hiesigen Umsetzung des Builder-Pattern liegt eine Basis-Abstraktion zugrunde, die dafür zuständig ist, die korrekte Vorlage für ein Eingabeelement in das Programm zu laden und für den Einsatz in der Benutzerschnittstelle vorzubereiten. Ein konkreter Builder für ein Eingabeelement (wie z. B. der Schieberegler) bietet zudem Methoden an, die helfen den in Abbildung 4 vorgestellen Vertrag zwischen Datenursprung (SEE-City-Instanz) und Eingabeelement zu erfüllen.

5. Evaluation

Im Anschluss an die Implementierung fand die Nutzerstudie statt, deren Evaluation in diesem Kapitel behandelt wird. Ziel der Evaluation war die Bewertung der Nutzerfreundlichkeit der neugewonnenen Komponente für SEE.

Anhand des Wissens um die Nutzerfreundlichkeit der beiden Eingabemedien Desktop und VR soll die folgende Frage beantwortet werden:

Wird die Nutzerfreundlichkeit für verschiedene Eingabemedien unterschiedlich wahrgenommen und eingeschätzt, wenn dieselbe Implementierung benutzt wird?

Durch die Beantwortung der Frage wird auch bestätigt oder widerlegt, ob dieselbe Implementierung einer Benutzerschnittstelle ohne größere Anpassungen in der Lage ist, den Bedürfnissen von unterschiedlichen Eingabemedien gerecht zu werden, oder ob eine native Implementierung für jedes Eingabemedium erforderlich ist.

Im Folgen wird genauer auf die Nutzerstudie eingegangen. Dabei wird der gewählte Fragebogen (System Usability Scale (SUS)) vorgestellt, aber auch andere Fragebögen diskutiert.

5.1. Nutzerstudie

Der Ablauf der Nutzerstudie sieht zwei Durchläufe für die Proband:innen vor, die in ihrer Reihenfolge variieren, d. h. einige Proband:innen werden zunächst die Anwendung am Desktop bedienen und dann in VR und manche andersherum. In beiden Varianten starten die Proband:innen zunächst mit einer geöffneten Benutzerschnittstelle, da dies dem regulären Gebrauch am ehesten entspricht - der zielführende Einsatz der Software ohne geladenen Graphen ist nicht möglich. Das Ziel für die Proband:innen ist nun, eine Code-City nach ihren Belieben zu konfigurieren und dann zu laden beziehungsweise zeichnen zu lassen. Ist dies geschafft, schließt sich das Menu und der reguläre Programmablauf kann beginnen, was für diese Studie nicht von Belang ist. Sobald die Interaktion mit dem Menu beendet ist, können die Proband:innen entweder das Menu erneut öffnen und ihre Angaben korrigieren oder neue treffen oder sie beenden ihren Durchlauf.

Im Anschluss an einen jeden Durchlauf erhalten die Proband:innen ein Exemplar des Fragebogens und bewerten ihre Eindrücke. Auf den Fragebogen wird im Folgenden näher eingegangen.

5.2. Fragebogen - System Usability Scale

Viele Faktoren haben auf die Wahl des SUS (Brooke, 1995) als Fragebogen für diese Arbeit Einfluss gehabt. Aufgrund der Covid-19-Pandemie ist es nicht vertretbar, eine Studie mit vielen Teilnehmer:innen unter semiprofessionellen Bedingungen durchzuführen. Daher muss der gewählte Fragebogen bereits bei einer kleinen Stichprobenmenge zuverlässige Ergebnisse garantieren. Darüber hinaus muss der Fragebogen auch von allen Proband:innen gut verstanden werden können, angefangen mit der Dokumentensprache. Da alle geladenen Proband:innen Deutsch als Muttersprache sprechen, muss der Fragebogen ebenfalls in Deutsch formuliert sein, um mögliche Sprachbarrieren und Ungenauigkeiten zu vermeiden.

Wie bereits angedeutet, müssen die Proband:innen eine eindeutige Aufgabe erledigen: das Konfigurieren und Laden einer Code-City mittels der implementieren GUI. Der SUS-Fragebogen ist mit seinen zehn Fragen für dieses Anwendungsszenario konzipiert worden (Grier et al., 2013). Abbildung 9 zeigt den in dieser Studie verwendeten Fragebogen. Die zehn Fragen werden diskret mit Werten von eins bis fünf bewertet, wobei eins eine völlige Ablehnung der formulierten Aussage entspricht und fünf einer uneingeschränkten Zustimmung. Die Fragen werden abwechselnd positiv und negativ gestellt, weshalb die Werte vor der Zusammenrechnung erst normiert werden müssen. Das Ergebnis ist eine Zahl zwischen 0 und 100. Die Interpretation dieses Werts ist nicht standardisiert, verschiedene empirische Forschungen versuchen aber eine Standardisierung zu erwirken. Sauro und Lewis (2016) geben ein Beispiel, wie der Wertebereich des SUS auf eine Benotung von A-F (angelehnt an US-amerikanische Schulnoten) skaliert werden kann, um das Ergebnis greifbarer für diverse Stakeholder zu machen:

SUS Score Range	Grade	Percentile Range
84.1–100	A+	96–100
80.8–84.0	A	90–95
78.9–80.7	A–	85–89
77.2–78.8	B+	80–84
74.1–77.1	B	70–79
72.6–74.0	B–	65–69
71.1–72.5	C+	60–64
65.0 –71.0	C	41–59
62.7–64.9	C–	35–40
51.7–62.6	D	15–34
0.0–51.6	F	0–14

Abbildung 7: *Vereinfachen des SUS-Werts mittels Schulnoten*
Quelle: Sauro und Lewis, 2016

5.3. Fragebogen - Software Usability Measurement Inventory

Der SUS-Fragebogeni ist mit seinen zehn Fragen überschaubar im Vergleich zu dem 50 Fragen langen Software Usability Measurement Inventory (SUMI). Der SUMI-Fragebogen wurde Anfang der 1990er in der Human Factors Research Group der Universität College Cork in Irland entworfen. Während der SUS-Fragebogen lediglich eine Metrik für die Benutzerfreundlichkeit liefert, teilt sich das Ergebnis des SUMI in fünf Metriken: Effizienz des Systems, emotionale Reaktion auf das System, Grad der Selbsterklärung und Dokumentation der Software, das Kontrollgefühl der Proband:innen und die Erlernbarkeit. Der SUS und der SUMI haben gemein, dass beide *post-study* (oder auch *post-test*) Fragebögen sind (Sauro & Lewis, 2016). *post-study* Fragebögen werden im Gegensatz zu *post-task* Fragebögen am Ende des gesamten Testdurchlaufs zur Beantwortung gereicht. *post-task* Fragebögen wiederum schließen direkt an eine sehr bestimmte Aufgabe an, die die Proband:innen erledigen mussten, d. h. nach dem Erfüllen einer Aufgabe wird der Test pausiert, der Fragebogen wird ausgefüllt und der Test fährt mit etwaigen weiteren Fragen fort.

Die Benutzung des SUMI für diese Arbeit kam nicht in Frage, da eine Lizengebühr zu entrichten ist.

5.4. Fragebogen - Subjective Mental Effort Question

Ein Beispiel für einen *post-task* Fragebogen (siehe 5.3) ist der Subjective Mental Effort Question (SMEQ). Im Gegensatz zu den bisher genannten Fragebögen weist der SMEQ lediglich eine Frage auf, wobei eine bestimmte Aufgabe in ihrem mentalen Aufwand bewertet wird von 0 bis 150, wobei 0 bedeutet, dass die Aufgabe überhaupt nicht fordernd zu lösen war, während Werte über 110 bedeuten, dass die Aufgabe außerordentlichst anstrengend zu lösen ist (Sauro & Lewis, 2016). Abbildung 8 zeigt die gesamte Skala mit allen Labels. Da nicht einzelne Aufgaben wie beispielsweise "Auswahl einer Datei mittels Dateibrowser" gestellt wurden, sondern die Proband:innen gezielt animiert wurden, die Benutzerschnittstelle frei zu verwenden, kamen *post-task* Fragebögen für diese Arbeit nicht in Betracht.

5.5. Studienablauf

Die Covid-19-Pandemie erforderte besondere Maßnahmen, um die Sicherheit aller Beteiligten gewährleitsten zu können. Während es zunächst geplant war, mehrere Proband:innen für einen Tag einzuladen, stellte sich dies doch als zu großes Risiko heraus. Die Rückmeldung der ersten eingeladenen Proband:innen bestätigte dies. Ebenso wurde das Tragen einer Maske vorausgesetzt. Der Raum, in dem die Studie stattfand, wurde gut durchlüftet und die Gegenstände (Maus, Tastatur, VR-Brille usw.), mit denen die Proband:innen in Kontakt kamen, wurden nach Beendigung eines Ablaufs desinfiziert. Zudem hat sich die testbegleitende Person regelmäßig auf eine Infizierung testen lassen.

Nach dem Empfang der Proband:innen wurde ein kurzes Vorgespräch abgehalten, in dem den Proband:innen der Modus der Studie erläutert und das SEE-Projekt vorgestellt wurde. Nachdem mögliche erste Fragen geklärt wurden, begab sich die Testperson je nach Auslosung entweder an einen Schreibtisch, um den Desktop-Part zu durchlaufen oder stellte sich in die Mitte des Raums für den VR-Part. Dann wurde die Benutzerschnittstelle vorgestellt und technische Feinheiten wie beispielsweise Tastenkürzel erläutert. Die Proband:innen wurden nun gebeten, das Menu zu erkunden mit dem gröberen Ziel, eine Code-City zu laden. Nach Beendigung eines jeden Durchlaufs - sobald die Proband:innen zufrieden mit ihrer Arbeit waren - erhielt die Testperson einen Fragebogen und füllte diesen aus. Anschließend wurde der noch fehlende, andere Part durchlaufen und der Fragebogen erneut vorgelegt. Diesmal war es der Testperson auch möglich, freies, qualitatives Feedback in Form eines Freitextfelds zu geben. Im Anschluss wurden die Proband:innen verabschiedet.

Es gab keine Zwischenfälle.

5.6. Ergebnisse

Nach dem der Fragebogen gewählt und die Nutzerstudie durchgeführt wurde, gilt es nun die Ergebnisse vorzustellen: zunächst die Darstellung der Rohdaten und dann ein Test auf Signifikanz der Daten.

Im folgenden werden die Ergebnisse des A (Desktop) und B (VR) Bogen vorgestellt.

#	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10
p1	4	2	5	1	4	2	5	2	5	2
p2	3	2	4	3	4	1	4	2	4	2
p3	5	3	4	3	5	1	5	2	4	3
p4	3	1	4	5	4	1	4	1	2	2
p5	4	2	4	3	4	1	3	2	2	3
p6	3	2	4	2	3	2	4	1	4	3
p7	4	2	5	3	2	4	4	1	2	1
p8	4	1	5	2	4	3	4	1	2	1

Tabelle 1: *Die Angaben der Proband:innen für die 10 Fragen nach dem Desktop-Durchlauf.*

#	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10
p1	5	1	4	2	5	2	4	1	5	2
p2	4	2	4	3	4	2	4	3	4	3
p3	4	2	4	3	4	1	4	3	3	3
p4	3	2	3	4	2	1	5	3	3	2
p5	4	2	4	1	3	3	4	3	4	2
p6	2	3	3	2	3	3	4	1	3	2
p7	3	1	5	2	2	3	3	3	4	1
p8	5	1	4	2	1	1	4	1	3	3

Tabelle 2: *B: Die Angaben der Proband:innen für die 10 Fragen nach dem VR-Durchlauf.*

Damit ergibt sich für den Desktop-Durchlauf ein SUS-Score von 72,5. Das entspricht nach den Kenntnissen aus Kapitel 5.2 also einer Benotung mit C+, sprich 3+ im deutschen Schulnotensystem - eine durchaus befriedigende Nutzungserfahrung. Für den VR-Durchlauf berechnet sich ein SUS-Score von 69,1, was einem C (einer

3) gleichtkommt. Die Benutzerfreundlichkeit wird also im VR-Setting negativer wahrgenommen als am Desktop.

Im folgenden wird die Signifikanz dieser Erhebung beurteilt. Dafür leiten sich aus der gestellten Forschungsfrage folgende Null- und Gegenhypothese ab (die Auswertung nutzt das 90%-Konfidenzintervall).

$\boldsymbol{H_0}$: Die Benutzerfreundlichkeit der Schnittstelle wird zwischen den Eingabemedien Desktop und VR als gleich wahrgenommen.

$\boldsymbol{H_1}$: Die Benutzerschnittstelle der Schnittstelle wird zwischen den Eingabemedien Desktop und VR unterschiedlich wahrgenommen.

Für einen solchen zweiseitigen Hypothesentest bieten sich verschiedene Tests an, die sich durch die zugrunde liegenden Annahmen über die Stichprobe unterscheiden. Der Wilcoxon-Vorzeichen-Rang-Test beispielsweise ist ein nichtparametrischer Test, der keine Normalverteilung des Datensatzes voraussetzt, während ein abhängiger t-Test eine Normalverteilung für Stichproben ($n < 50$) voraussetzt (Bortz, 2010). Daher bietet es sich an, zunächst einen Normalitätstest durchzuführen, bevor unter falschen Annahmen einer parametrischer Test durchgeführt wird. Aufgrund der kleinen Stichprobengröße bietet sich der Anderson-Darling-Test an (Berna Yazici & Senay Yolacan, 2007). Dem Test werden die Differenzen der SUS-Score-Paare zugrundegelegt:

#	**Desktop**	**VR**	**Differenz**
p1	85	87.5	-2.5
p2	72.5	67.5	5
p3	77.5	67.5	10
p4	67.5	60	7.5
p5	65	70	-5
p6	70	60	10
p7	65	67.5	-2.5
p8	77.5	72.5	5

Tabelle 3: *Die SUS-Scores der einzelnen Personen am Desktop und in VR, plus die dazugehörigen Differenz.*

Die Hypothesen für diesen Test lauten:

$\boldsymbol{H_{N0}}$: Die Stichprobe entspricht einer Normalverteilung.

$\boldsymbol{H_{N1}}$: Die Stichprobe entspricht nicht einer Normalverteilung.

Für die vorliegenden Daten ergibt sich ein p-Wert von 0.1873. Dieser Wert is größer als die eingangs erwähnten $\alpha = 0.10$, weshalb die Nullhypothese H_{N0} nicht abgelehnt werden kann. Ausgehend davon wird im Folgenden ein abhängiger t-Test durchgefürt, um die Signifikanz der Stichproben beurteilen zu können.

Ein abhängiger t-Test (Paardifferenztest) wird realisiert, indem ein regulärer t-Test über die Differenz der SUS-Score-Paare durchgefürt wird. Hier ergibt sich ein p-Wert von 0.1472, der ebenfalls größer als die gewählten $\alpha = 0.10$ sind. Dadurch lässt sich die Nullhypothese H_0 nicht ablehnen und weshalb zu urteilen ist, dass sich die Benutzerfreundlichkeit zwischen dem Einsatz am Desktop und in VR nicht signifikant unterscheidet.

6. Ausblick

Die Evaluation der Schnittstelle hat nicht nur Ergebnisse mittels des SUS-Fragebogen erbracht, sondern den Proband:innen stand es auch frei, ihre Nutzungserfahrung, Kritik und Vorschläge am Ende der Studie frei zu formulieren.

Einer der am häufigsten genannten Punkte zielt weniger auf die Bedienbarkeit der Schnittstelle, sondern eher auf das Verständnis der vorliegenden Gesamtsoftware ab: die einzelnen Werte, die sich mit der Schnittstelle manipulieren lassen, waren den Proband:innen zuweilen nicht ausreichend erklärt. Es wurde vorgeschlagen, beispielsweise über einen *Tooltip* an jedem Eingabeelment, Erklärtexte für die einzelnen Werte einzubringen. So sahen sich einige Proband:innen gezwungen, Rückfragen zu stellen.

Während die Bedienung am Desktop mittels Maus und Tastatur vielen vertraut vorkam - besonders wenn die Personen einen Hintergrund mit Videospielen hatten -, so haben die Proband:innen ein bestimmtes Eingabeelemnt besonders im VR-Umfeld kritisiert beziehungsweise als benutzerunfreundlich beschrieben: der Schieberegler (*range slider*). Es erfordert Fingerspitzengefühl, mittels VR-Controller-Steuerung den Regler zu greifen und zu bewegen. Vorschläge zur Verbesserung sind hier z. B. das Vergrößern des Reglers, sobald mit dem VR-Zeiger über das Eingabeelment gezeigt wird. Auf diese Art ließe sich der Regler leichter greifen. Doch auch allgemein wurde die Tauglichkeit von Schiebereglern angezweifelt, da diese per Definition Schranken um den möglichen Wertebereich setzen. Innerhalb dieses Wertebereichs kann ein beliebiger Wert gewählt werden, außerhalb ist es nicht möglich. Für bestimmte Eingabewerte ist dies gewiss die richtige Wahl (z. B. Prozentzahlen), aber für Felder, deren Wert frei gewählt werden soll, gibt es bessere Alternativen. Die Nutzerstudie hat gezeigt, dass die Spracherkennung mittels der Microsoft Windows Speech-Recognition-API sehr gut funktioniert und auch Dezimalzahlen beliebiger Längen korrekt interpretieren kann. Daher kann der Schieberegler an Stellen ohne eigentlichen Wertebereich durch ein reguläres Zahleneingabefeld ersetzt werden, das am Desktop per Tastatur und in VR per Sprache befüllt wird.

Ein Punkt, der in einigen Fällen für Verwirrung gesorgt hat, ist das Fehlen des Scrollbalkens in den einzelnen Menu-Seiten. Dies ist auf einen Fehler zurückzuführen, der bis zum Abschluss dieser Arbeit nicht ergründet werden konnte. Ohne einen Indikator, dass an einer bestimmten Stelle weiter Inhalt angezeigt werden kann, wird der besagte Inhalt schnell übersehen. Gerade wenn die Nutzenden nicht mit jedem Eingabewert vertraut sind, können sie auch keinen Wert vermissen, was zu einer verfälschten Benutzererfahrung mit der gesamten Software führen kann.

In dieser Arbeit wurde keine Unterstützung für die Bedienung mittels Gamepad oder Microsoft HoloLens implementiert - beides Systeme, die grundsätzlich von SEE unterstüzt werden. Das Ergebnis dieser Arbeit lässt aber vermuten, dass die Bedienung auch mit anderen Eingabemedien ohne größere Anpassungen möglich sein sollte.

Anhang

Anhangsverzeichnis

A. Abbildungen

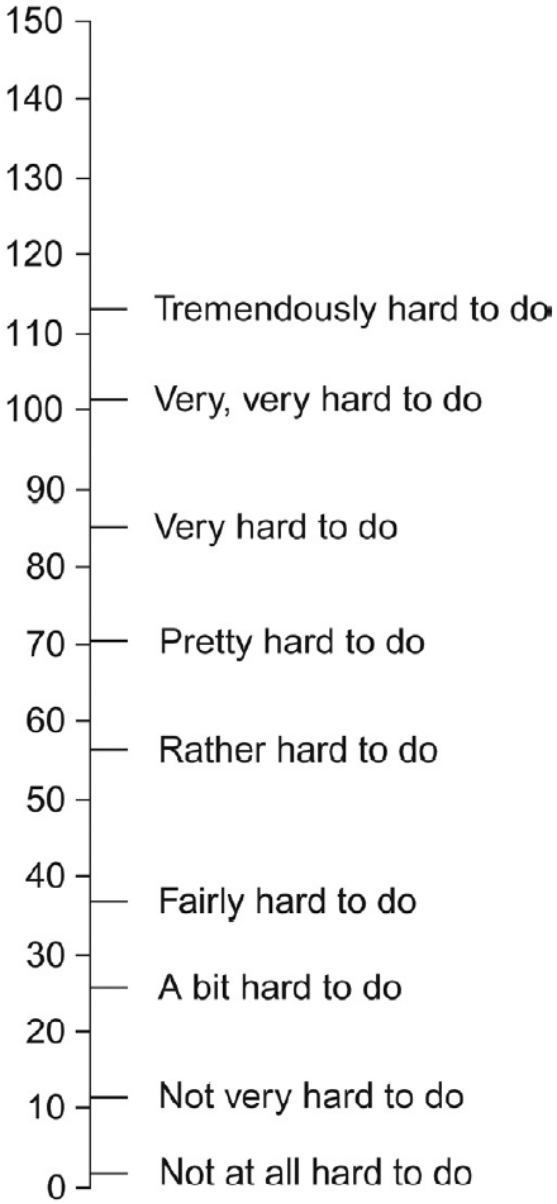

Abbildung 8: *Die SMEQ-Skala*
Quelle: Sauro und Lewis, 2016

Fragebogen zur System-Gebrauchstauglichkeit

1. Ich denke, dass ich das System gerne häufig benutzen würde.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

2. Ich fand das System unnötig komplex.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

3. Ich fand das System einfach zu benutzen.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

4. Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

5. Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

6. Ich denke, das System enthielt zu viele Inkonsistenzen.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

7. Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

8. Ich fand das System sehr umständlich zu nutzen.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

9. Ich fühlte mich bei der Benutzung des Systems sehr sicher.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

10. Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.

Stimme überhaupt nicht zu				Stimme voll zu
1	2	3	4	5
○	○	○	○	○

Abbildung 9: *Der SUS-Fragebogen übersetzt ins Deutsche von SAP*
Quelle:
https://blogs.sap.com/2016/02/01/system-usability-scale-jetzt-auch-auf-deutsch/

Literatur

Berna Yazici & Senay Yolacan. (2007). A comparison of various tests of normality. *Journal of Statistical Computation and Simulation*, *77*(2), 175–183.

Bortz, J. (2010). *Verteilungsfreie Methoden in der Biostatistik*. Springer.

Brooke, J. (1995). SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, *189*.

Carroll, J. M., Mack, R. L. & Kellogg, W. A. (1988). Interface Metaphors and User Interface Design. *Handbook of Human-Computer Interaction* (S. 67–85). Elsevier.

Gamma, Erich and Helm, Richard and Johnson, Ralph and Vlissides, John M. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

Grier, R. A., Bangor, A., Kortum, P. & Peres, S. C. (2013). The System Usability Scale. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *57*(1), 187–191.

Hassenzahl, M. & Tractinsky, N. (2006). User experience - a research agenda. *Behaviour & Information Technology*, *25*(2), 91–97.

Köhler, W. (1970). *Gestalt psychology: An introduction to new concepts in modern psychology / by Wolfgang Köhler*. Liveright.

Koschke, R. (2020). Auge in Auge mit Ihrer Softwarearchitektur.

Panas, T., Berrigan, R. & Grundy, J. (2003). A 3D Metaphor for Software Production Visualization. *International Conference on Information Visualization*, 314.

Sauro, J. & Lewis, J. R. (2016). *Quantifying the user experience: Practical statistics for user research* (Second edition). Morgan Kaufmann.

Stone, D. L. (2005). *User interface design and evaluation*. Morgan Kaufmann.

van Welie, M., van der Veer, G. C. & Eliëns, A. (2001). Patterns as Tools for User Interface Design. *Tools for Working with Guidelines* (S. 313–324). Springer London.

Wettel, R. & Lanza, M. (2007). Visualizing Software Systems as Cities. *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*.

Yu, S. & Zhou, S. (2010). A survey on metric of software complexity. *2010 2nd IEEE International Conference on Information Management and Engineering*.