OST
Ostschweizer
Fachhochschule

Lebendige Software Cities zur Visualisierung von Softwareprojekten

Studienarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Frühjahrssemester 2022

Autoren: Benny Joe Villiger und Thomas Zahner

Betreuer: Prof. Dr. Frieder Loch

Abstract

Durch die zunehmende Digitalisierung und der damit verbundenen Nachfrage nach Software, nimmt auch die Bedeutung deren Qualität und Sicherheit zu. Daher ist es wichtig die Softwarequalität regelmässig zu überprüfen und zu verbessern. Dieser Prozess kann durch eine Vielzahl an Werkzeugen unterstützt werden.

Das Ziel der vorliegenden Arbeit ist es, den Zustand und die Qualität eines Softwareprojekts sinnvoll und intuitiv zu visualisieren. Die Visualisierung wird mittels einer “Stadtmetapher” umgesetzt. Dies bedeutet, dass der Zustand des Softwareprojekts, durch die Abbildung einer Stadt widerspiegelt wird. Die Messung der Qualität erfolgt durch ein statisches Codeanalysewerkzeug, das die notwendigen Metriken zur Verfügung stellt. Die Visualisierung wird mittels virtueller Realität (VR) realisiert.

In einem ersten Schritt wurden Werkzeuge zur statischen Codeanalyse evaluiert und anhand zuvor definierten Kriterien verglichen. SonarQube schnitt in diesem Vergleich am besten ab und wurde deshalb für den Einsatz in diesem Projekt ausgewählt. Basierend auf der vorhergehenden Arbeit wurde zur Visualisierung der Stadt die Webtechnologie Babylon.js verwendet. Babylon.js ist eine webbasierte 3D-Engine um Objekte und Grafiken in Echtzeit im Webbrowser darzustellen und verfügt über eine gute Integration mit VR-Geräten. In einem nächsten Schritt wurden Metaphern festgelegt, die definieren, wie die Metriken visualisiert werden.

Das Resultat ist eine Webapplikation, die es erlaubt visuell in ein Softwareprojekt einzutauchen. Dabei können potenzielle Qualitäts- und Sicherheitsprobleme auf eine spielerische Art entdeckt werden. Gegen Ende des Projekts wurden Benutzertests mit ausgewählten Benutzergruppen durchgeführt, um den Nutzen der virtuellen Stadt in der Praxis zu überprüfen und Rückmeldungen zu erhalten. Den Haupteinsatzzweck von Software Cities sehen die Testprobanden in der Ausbildung. Ausserdem konnten basierend auf den Rückmeldungen, Erweiterungsideen für eine Folgearbeit gesammelt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Management Summary

Ausgangslage

Das Ziel dieser Arbeit ist die Visualisierung von Metriken in einer VR-Applikation. Als Basis dient eine vorgängige Studienarbeit, bei der eine VR-Applikation erstellt wurde, mit der GitHub-Repositories als Stadt visualisiert werden können. Die Applikation verwendet die Technologie Web-XR zur Darstellung der Stadt. Die Entscheidung, Web-Technologien für diese Art von Applikation anzuwenden, soll zum Ziel führen, dass die Applikation plattformübergreifend verwendet werden kann. Das Projekt weist aber zu diesem Zeitpunkt mehrere nennenswerte Limitationen auf. Erstens wird nur die Analyse von GitHub-Repositories und Javaprojekten unterstützt. Zweitens ist die Darstellung der Stadt nur mittels VR-Geräten möglich, die Stadt kann also nicht direkt im Browser dargestellt werden. Drittens werden nur wenige einfache Metriken verwendet und visualisiert.

Vorgehen

Zu Beginn war es notwendig, sich in das vorhergehende Projekt einzuarbeiten und mögliche Erweiterungen und Verbesserungen auszuarbeiten. Auf die Einarbeitung folgte ein Refactoring des Codes, um den Code besser zu verstehen und das Projekt auf die zukünftigen Änderungen vorzubereiten. Des Weiteren stellte es sich als Vorteil heraus, dass man sich zu Beginn mit einem 3D-Modellierungsprogramm vertraut gemacht hat, dadurch konnten Änderungen an bestehenden Modellen vorgenommen werden können. Die Evaluierung eines Tools für die statische Codeanalyse sollte auch zu Beginn erfolgen, da ein Grossteil des vorliegenden Projekts darauf basiert. Sobald die Vorbereitungsarbeiten erfolgt waren, konnte mit der Entwicklung des Backends begonnen werden, dieses wird dazu verwendet, die Metriken der Codeanalyse über eine API dem Frontend zur Verfügung zu stellen. Anschliessend wurde die Visualisierung der Modelle mit Babylon.js implementiert und dokumentiert. Nach Abschluss der Implementation wurden Benutzertests durchgeführt, um den tatsächlichen Nutzen des Projekts zu erfahren, und Verbesserungsvorschläge zu erhalten.

Ergebnisse

Durch die investierte Zeit in die Planung und Einarbeitung und dank der vorhergehenden Arbeit, konnte schon früh mit der Implementation begonnen werden. Die zuvor erwähnten Limitationen konnten alle erfolgreich beseitigt werden. Die Nutzertests zeigten auf, dass die Applikation durchaus sinnvolle Anwendungszwecke hat und erlaubten es, Verbesserungsvorschläge und Ideen zu sammeln.

Danksagung

Wir bedanken uns bei Prof. Dr. Frieder Loch für die tatkräftige Unterstützung, die verschiedenen Inputs und die aufgewendete Zeit fürs Korrekturlesen, während der Projektlaufzeit.

Des Weiteren bedanken wir uns bei den Testpersonen, ohne die es nicht möglich gewesen wäre, ein Fazit für die Nutzerforschung zu ziehen.

1 Einführung

1.1 Ziel der Arbeit

Ziel der Arbeit ist es, eine funktionsfähige VR-Applikation zu entwickeln, mit der Git Repositories analysiert und anschliessend visualisiert werden können.

1.1.1 Metriken

Die Stadt soll ihre Darstellung anhand von Softwaremetriken verändern. Wir möchten die Qualität vom Code anhand von Metriken messen, und diese in der Stadt durch Metaphern darstellen.

1.1.2 Bestehende Limitationen beseitigen

Die Arbeit basiert auf einer vorhergehenden Arbeit, die zum Teil limitierten Funktionsumfang hat. Diese bestehenden Limitationen möchten wir in unserer Arbeit beseitigen.

Limitation GitHub API: Die Applikation kann nur mit öffentlichen GitHub Repositories umgehen. Wir möchten die Funktion insofern erweitern, dass man mit verschiedenen Git Providern arbeiten kann.

Limitation Programmiersprache: Bisher kann nur die Qualität von Java Code gemessen werden. Wir möchten diese Funktion erweitern, sodass die Qualität der am häufigsten verwendeten Programmiersprachen gemessen werden kann.

Limitation VR-Modus: Zu Beginn des Projektes ist es nur möglich die Stadt mittels virtueller Realität zu erkunden. Es soll neu die Möglichkeit geben die Stadt im Webbrowser darzustellen.

1.2 Beschreibung der vorhergehenden Arbeit

In der vorhergehenden Arbeit (Hindermann und Hirzel 2022a) sollte die Frage beantwortet werden, wie gut sich die virtuelle Realität mit Webtechnologien umsetzen lässt. Es wurden verschiedene Tools und Frameworks miteinander verglichen, um dann mithilfe der Implementation eines Prototyps ein Fazit zu ziehen. Die Testapplikation beschäftigt sich mit einer unkonventionellen Art Software zu visualisieren. Die Idee war es, Softwareprojekte in Form von Software Cities darstellen zu können.

2 Begriffserklärung und Anwendungsfälle

Ziel dieses Kapitels ist es, einen Überblick zu den Begriffen aus den Bereichen des Software-Engineerings und der virtuellen Realität zu verschaffen.

2.1 Softwarequalität

Das Konsortium für Information und Softwarequalität (CISQ) schätzte im Jahr 2018 die Kosten durch schlechte Softwarequalität auf 2,84 Billionen US-Dollar, alleine in den USA. Die Kosten sind hauptsächlich zurückzuführen auf Verluste durch Softwareausfälle, Problemen mit Legacy-Systemen, technische Schulden, sowie auf das Finden und Beheben von Defekten. (Krasner 2018)

2.1.1 McCall's Modell der Softwarequalität

Das McCall Modell ist eine Möglichkeit, die Qualität von Software zu definieren. Es wurde 1977 entwickelt und besteht aus Qualitätsfaktoren und Qualitätskriterien. (Samadhiya, Wang, und Chen 2010)

Die drei Produkt-Qualitätsfaktoren, *Product Operation*, *Product Revision* und *Product Transition*, bestehen aus vordefinierten Software-Qualitätsfaktoren, die in Abbildung 1 dargestellt sind.

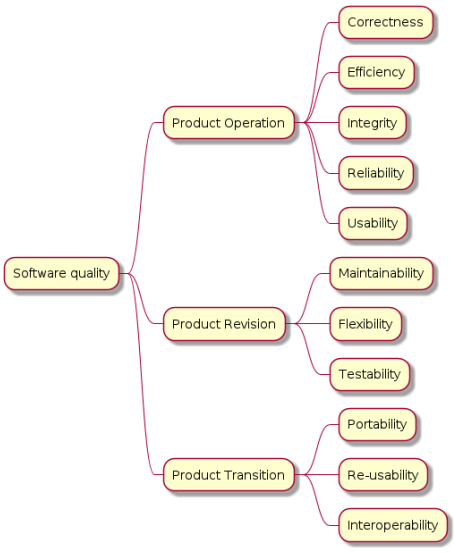

Abbildung 1: McCall Software Quality Model

Die Qualitätsfaktoren bestehen wiederum aus Qualitätskriterien, die sich jedoch, abhängig von den Anforderungen der jeweiligen Software, unterscheiden können. Qualitätsfaktoren können, im Gegensatz zu Qualitätskriterien, direkt gemessen werden. Messungen von Qualitätsfaktoren erfolgen unterschiedlich. Zum Beispiel werden Qualitätsfaktoren im Bereich *Usability* mittels Usability-Tests mit ausgewählten Testgruppen durchgeführt, während *Efficiency* Anforderungen meist programmatisch auf einem Testsystem durchgeführt werden. Viele (aber nicht alle) Qualitätsfaktoren können durch statische Codeanalysen getestet werden.

2.2 Codequalität

Codequalität und Softwarequalität sind nicht dasselbe. Codequalität hat aber auf jeden Fall Einfluss auf einzelne Qualitätsfaktoren, die wiederum die Softwarequalität beeinflussen.

2.2.1 Messung von Codequalität

Codequalität kann mittels statischer Codeanalyse gemessen werden. Statische Codeanalysewerkzeuge ermöglichen es Quellcode zu analysieren, ohne diesen auszuführen und Codequalität zu messen. Die Werkzeuge können verschiedene Eigenschaften, wie *Komplexität*, *Wiederverwendbarkeit* und *Duplikation*, des Quellcodes analysieren und erfassen. Diese einzelnen Eigenschaften werden Metriken genannt und können beliebig komplex zu erfassen sein.

Der Zweck von Softwaremetriken ist es, Bewertungen, über das Einhalten von Qualitätsanforderungen während eines Softwarelebenszyklus machen zu können. Das Benutzen von Metriken reduziert dabei die Subjektivität in einem Qualitätsassessment und die Kontrolle der Softwarequalität, indem man eine quantitative Basis, um Entscheidungen zu treffen, zur Verfügung stellt. („IEEE Standard for a Software Quality Metrics Methodology“ 1998)

Eine Metrik ist daher eine (meist mathematische) Funktion, die eine Eigenschaft als Zahlenwert abbildet. Durch das Sammeln von Metriken können Vergleichs- oder Bewertungsmöglichkeiten geschaffen werden.

2.3 Virtuelle Realität

Virtuelle Realität, auch abgekürzt als VR, hat in den vergangenen Jahren zunehmend an Bedeutung gewonnen. Die Anwendungsfälle von VR sind vielfältig und reichen vom Einsatz für die Unterhaltung über Bildungszwecke bis hin zur Anwendung für professionelle Zwecke. Beispielsweise können Feuerwehrleute mit VR Technologie trainiert werde, ohne dass sie sich in Gefahr begeben müssen. (Nell 2020) Im Gleichschritt mit der Zunahme an Bedeutung während den letzten Jahren entwickelte sich auch die Technologie als Ganzes weiter.

Doch was genau versteht man eigentlich unter VR?

> Unter virtueller Realität versteht man den Einsatz von Computertechnologien, um den Eindruck einer interaktiven dreidimensionalen Welt zu erwecken, in der die Objekte ein Gefühl der räumlichen Präsenz vermitteln. Unter “räumlicher Präsenz” versteht man hierbei, dass die Objekte in der Umgebung eine Position im dreidimensionalen Raum relativ zur Person haben. Der Hauptunterschied zwischen konventioneller dreidimensionaler Computergrafik und virtueller Realität ist, dass man in der virtuellen Realität mit der Sache selbst arbeitet anstatt mit Bildern von der Sache. (NASA Advanced Supercomputing, o. J.)

Es ist anzumerken, dass es dabei nicht nur “die eine Art” virtueller Realität gibt. Es existieren verschiedene Formen der virtuellen Realität, in denen unterschiedlich viel Virtualität und Realität eingebracht werden. Aus diesem Zusammenhang leitet sich ein “Realität-Virtualität-Kontinuum” ab.

Wie sich in Abbildung 2 erkennen lässt, existiert auf der einen Seite die reale Umgebung und auf der anderen Seite die komplett virtuelle und simulierte Umgebung. Dazwischen befindet sich eine Mischung zwischen Realität und Virtualität. Milgram et al. siedeln dazwischen die Begriffe *erweiterte Realität* und *erweiterte Virtualität* an und fassen das Zwischenspektrum als *gemischte Realität* zusammen. Da die Übergänge fliessend sind und es keine klaren

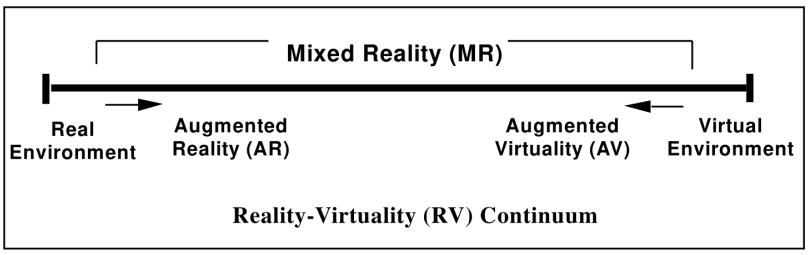

Abbildung 2: Realität-Virtualität-Kontinuum (Milgram u. a. 1994)

Abgrenzungen gibt, werden die Begriffe oftmals auch vertauscht oder als Synonyme verwendet. Dies ist auch ein Mitgrund, weshalb sich der Überbegriff *XR* in den letzten Jahren zusätzlich durchgesetzt hat.

Der Begriff XR umfasst alle Arten der Realität und bedeutet somit gleichzeitig AR (erweiterte Realität), VR (virtuelle Realität) und MR (gemischte Realität). Auch die grossen Technologieunternehmen wie Microsoft, Samsung, Apple, Google und Facebook machten in jüngster Zeit Gebrauch von diesem Überbegriff. Zusätzlich haben die Unternehmen verschiedene Vorstellungen von der Umsetzung. Die einen verwenden ein mobiles Display, andere eine smarte Brille und wieder andere ein VR-Headset. Einige Geräte unterstützen drei Freiheitsgrade, andere wiederum sechs. Das Technologiekonsortium Khronos Group hat eine Initiative namens OpenXR ins Leben gerufen. Damit wird eine offene gemeinsame Schnittstelle geschaffen, welche es unter anderem ermöglicht Apps zu entwickeln, die nicht so strikt in eine Kategorie eingeordnet werden können. (Goode 2019)

Neben dem OpenXR Standard entwickelte sich auch die Web-XR Spezifikation, welche das Ziel hat eine offene Schnittstelle für XR-Geräte im Webumfeld zu schaffen.

2.4 Immersion und Bildung

Immersion und virtuelle Realität können Lernprozesse unterstützen und Anwendungszwecke in der Bildung haben. Im Folgenden ein Auszug aus einer Studie von Microsoft:

> Immersion bezieht sich auf den geistigen Zustand, in dem man vollständig vertieft ist oder mit einer Aktivität beschäftigt ist. Immersion kann ein leistungsfähiges Mittel für Wissenstransfer und Identitätstransfer sein. Beides sind entscheidende Faktoren für einen grösseren Lernerfolg. Dieses Prinzip wird schon seit längerem beim spielbasierten Lernen angewandt, bei dem es darum geht, Identitäten zu übernehmen und mit ihnen zu spielen, sodass der Lernende echte Wahlmöglichkeiten hat. Die Lernenden erhalten die Möglichkeit, die Übertragung des Gelernten auf spätere Probleme zu üben. (Bonasio 2019)

2.5 Softwarevisualisierung mit virtueller Realität

Visualisierung ist nötig in der heutigen Zeit der Generierung von Masseninformationen. Visualisierung ermöglicht es grosse unüberschaubare Datenvolumen in etwas Überschaubares umzuwandeln. Virtuelle Realität ist ein Weg diese Visualisierungen zu ermöglichen und bietet ausserdem viele Vorteile gegenüber herkömmlichen Visualisierungssystemen. Durch die Anwendung von drei Dimensionen und geeigneten Metaphern können Informationslandschaften geschaffen werden, welche die Nutzer nach Belieben ansehen und erkunden können. Softwarevisualisierung, eine Untergruppe der Informationsvisualisierung, gibt es schon viele Jahre in unterschiedlichen Formen. Das Problem ist aber, dass in diesem Bereich immer noch Knoten- und Bogenstrukturen in zwei Dimensio-

nen verwendet werden, um immer grössere und komplexere Softwaresysteme zu visualisieren. (Knight und Munro 2000)

2.6 Software-City

Es existieren bereits Möglichkeiten, um dreidimensionale Visualisierungen von Software zu erstellen. Viele Visualisierungen scheitern aber in der Aufgabe, die relevanten Informationen des Systems darzustellen und tragen deshalb nicht zum besseren Programmverständnis bei. Dies ist häufig deshalb der Fall, weil die zusätzliche dritte Dimension dazu genutzt wird, noch mehr Informationen zu kommunizieren, was schnell zu Informationsüberflutung und Navigationsproblemen führt. Software wird oft als ein Graph von Knoten und Kanten dargestellt, der im 3D Raum schwebt, worin die Nutzer sich frei bewegen können. Diese Darstellung und zusätzliche Freiheit sind aber oftmals nicht intuitiv und führen zu Desorientierung und damit zu einer negativen Wahrnehmung von 3D Visualisierungen. Um dem entgegenzuwirken wird eine gute Metapher gebraucht, denn diese erlaubt es, Elemente in einen dem Benutzer vertrauten Kontext einzubetten. (Wettel und Lanza 2007)

Ein Beispiel einer Metapher ist die Darstellung als Sonnensystem. Bei dieser Metapher wird jedem Package eine Sonne zugeordnet, welche von Planeten umkreist wird. Jeder Planet stellt eine Klasse oder Datei dar und widerspiegelt auch deren proportionale Grösse. (Graham, Yang, und Berrigan 2004)

Eine noch gängigere Metapher um Software zu visualisieren ist die Stadtmetapher. Bei dieser werden verschiedene Stadtviertel dargestellt, um die Packages zu repräsentieren. Die Gebäude in einem Viertel entsprechen den Klassen im Package. Der Hauptgrund für die Bekanntheit dieser Metapher ist darauf zurückzuführen, dass die meisten Personen mit dem Stadtkonzept vertraut sind. (Wettel und Lanza 2007) Ausserdem bietet die Metapher auch eine hohe Flexibilität und Anpassungsmöglichkeiten bei der Darstellung einzelner Häuser. Die Häuser können unterschiedlichste Formen und Farben annehmen, wodurch dem Benutzer noch weitere Informationen vermittelt werden können.

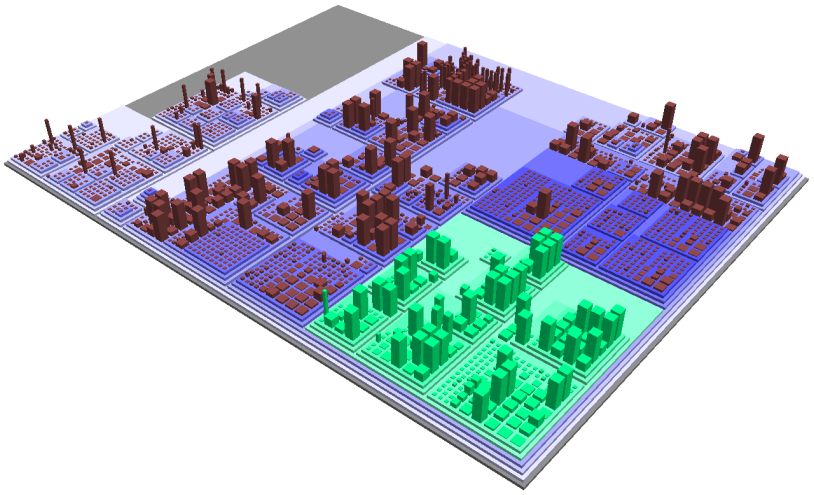

Abbildung 3: Visualisierung einer Software-City (Wettel und Lanza 2007)

In Abbildung 3 ist eine beispielhafte Software-City ersichtlich. Die Gebäude sind als unterschiedlich hohe Balken dargestellt. Auch zu erkennen sind verschiedene Stadtteile und Quartiere. Ein Stadtteil wurde selektiert und ist deshalb grün markiert.

3 Funktionale Anforderungen

Die funktionalen Anforderungen an dieses Projekt sind der Aufgabenstellung entnommen und wurden zum Zweck der Übersicht in diesem Kapitel als Use Cases im *Fully-Dressed Format* anhand der Methode von Larman (Larman 2004) beschrieben. Einzelne Use Cases, die bereits im Vorprojekt (Hindermann und Hirzel 2022a) umgesetzt sind, wurden an unsere Aufgabenstellung angepasst oder erweitert.

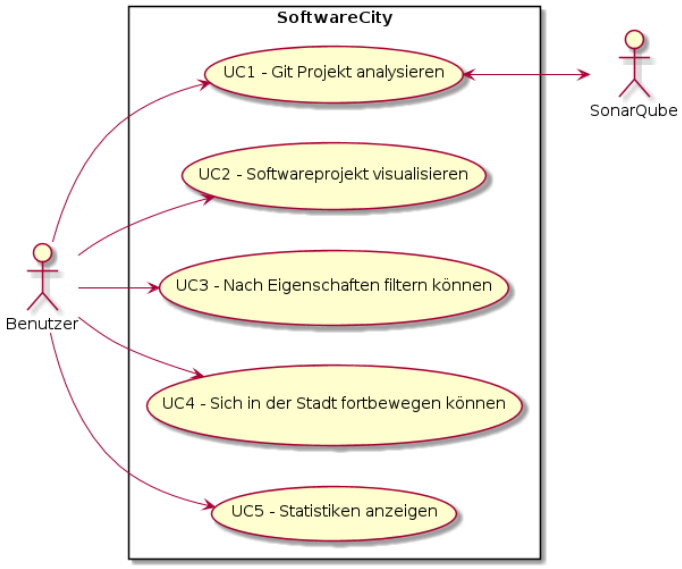

Abbildung 4: Use Case Diagramm

3.1 UC 1 Git Projekt analysieren

Anpassung und Aufteilung des UC1 - Übersicht über Softwareprojekt erhalten

Level

User-goal

Primary Actor

Benutzer

Stakeholders and Interests

Der Benutzer möchte eine Analyse für sein Projekt durchführen.

Preconditions

Der Benutzer hat sein Projekt in einem Git Repository abgespeichert, und kann einen öffentlichen Link zum klonen bereitstellen.

Success Guarantee

Der Benutzer konnte das Projekt erfolgreich analysieren und hat eine ID erhalten, womit der Analyseprozess in Zukunft umgangen werden kann.

Main Success Scenario

1. Der Benutzer besucht die Website der Software City.
2. Der Benutzer gibt den Link zu seinem öffentlichen Repository in die dazu passende Textbox ein und klickt anschliessend, auf den Knopf, um die Analyse zu starten.
3. Der Analyse Prozess wurde gestartet.
4. Die Analyse wurde erfolgreich durchgeführt, es wird eine ID angezeigt, mit der der Benutzer in Zukunft diesen Schritt überspringen kann. Zusätzlich erscheint ein Knopf, mit dem der Benutzer seine Stadt anschauen kann.

Alternative Paths

Alternative Path 1

2A1) Der Benutzer gibt einen Link zu einem nicht öffentlichen Git Repository (oder allgemein ein inkorrekter Link) ein und klickt anschliessend auf den Knopf, um die Analyse zu starten.

2A2) Der Analyse Prozess wird nicht gestartet, eine Fehlermeldung wird angezeigt und das Link Eingabefeld wird wieder angezeigt.

Alternative Path 2

4A1) Die Analyse war nicht erfolgreich. Eine Fehlermeldung wird angezeigt.

Frequency of Occurrence

Jedes Mal.

3.2 UC 2 Softwareprojekt visualisieren

Anpassung und Aufteilung des UC1 - Übersicht über Softwareprojekt erhalten.

Level

User-goal

Primary Actor

Benutzer

Stakeholders and Interests

Der Benutzer möchte eine Visualisierung seines Softwareprojekts betrachten.

Preconditions

1. Erfolgreiche Durchführung von UC 1.
2. Optional (aber für Main Success Scenario benötigt): Der Benutzer hat eine unterstützte VR-Brille eingerichtet und mit dem PC verbunden.

Success Guarantee

Der Benutzer konnte eine Visualisierung seines Softwareprojekts betrachten.

Main Success Scenario

1. Der Benutzer klickt auf den Knopf “Stadt besuchen”.
2. Ein “Enter-VR” Knopf erscheint.
3. Der Benutzer klickt auf den “Enter-VR” Knopf.
4. Die Software City wird auf der verbundenen VR-Brille dargestellt.

Alternative Paths

1A1) Der Benutzer klickt auf den Knopf “Stadt besuchen” (hat aber keine VR-Brille verbunden).

1A2) Es erscheint kein “Enter-VR” Knopf, die Software City wird direkt im Browser angezeigt.

Frequency of Occurrence

Regelmässig

3.3 UC 3 Nach Eigenschaften filtern können

Anpassung aus UC2 - Nach Eigenschaften filtern können.

Level

User-goal

Primary Actor

Benutzer

Stakeholders and Interests

Der Benutzer möchte die nur Gebäude, die gewisse Kriterien erfüllen, angezeigt bekommen.

Preconditions

Erfolgreiche Durchführung von UC2.

Success Guarantee

Der Benutzer konnte die Stadt erfolgreich nach gewissen Kriterien filtern.

Main Success Scenario

1. Der Benutzer will die Qualität der Software begutachten.
2. Der Benutzer gibt an, nach welcher Metrik er filtern will und definiert einen eigenen Schwellwert.
3. Die Gebäude, die den Schwellwert überschreiten, werden markiert.

Frequency of Occurrence

Selten

3.4 UC 4 Sich in der Stadt fortbewegen können

Anpassung aus UC6 - Sich in der Stadt fortbewegen können.

Level

User-goal

Primary Actor

Benutzer

Stakeholders and Interests

Der Benutzer möchte durch die Stadt gehen, und die Stadt von einer anderen Perspektive betrachten können.

Preconditions

Erfolgreiche Durchführung von UC2.

Success Guarantee

Der Benutzer konnte sich erfolgreich durch die virtuelle Stadt fortbewegen.

Main Success Scenario

1. Der Benutzer öffnet das Hauptmenü der Applikation.
2. Der Benutzer wählt “Stadt besichtigen”.
3. Der Benutzer markiert die Stelle in der Stadt, an der er sich gerne befinden möchte.
4. Der Benutzer wird an die gewünschte Stelle innerhalb der Stadt teleportiert und kann sich nun frei bewegen. Die Interaktionen mit den Gebäuden werden nach wie vor unterstützt.

Frequency of Occurrence

Regelmässig

3.5 UC 5 Statistiken anzeigen

Level

User-goal

Primary Actor

Benutzer

Stakeholders and Interests

Der Benutzer möchte die Metriken zu einzelnen Klassen anschauen.

Preconditions

1. Erfolgreiche Durchführung von UC 2.
2. Der Benutzer hat eine unterstützte VR-Brille eingerichtet und mit dem PC verbunden.

Success Guarantee

Der Benutzer konnte mit den Gebäuden in der Stadt interagieren, und kann die Metriken zu den einzelnen Gebäuden sehen.

Main Success Scenario

1. Der Benutzer schwenkt den Cursor seines rechten VR-Controllers auf ein Gebäude.
2. Das Gebäude wird umrandet.
3. Der Benutzer klickt auf das Gebäude.
4. Über dem linken Controller öffnet sich ein Fenster mit den Statistiken zu dem angeklickten Gebäude.
5. Der Benutzer kann seinen linken Controller bewegen, dabei bewegt sich das Fenster mit.

Frequency of Occurrence

Regelmässig

4 Architektur

Die Applikation besteht aus Frontend-, Backend- und Analysekomponenten. Die Architektur des Frontends wurde weitgehend vom Vorprojekt übernommen und ist im folgenden Dokument ersichtlich: (Hindermann und Hirzel 2022a). Die Architektur der anderen Komponenten ist in diesem Kapitel beschrieben.

4.1 Refactoring

Das Vorprojekt hatte bereits eine Art, Metriken von Code zu messen. Diese wurde jedoch nur prototypartig implementiert und war von der GitHub API abhängig. Alle verwendeten Klassen und Metriken wurden genau auf diesen Use Case zugeschnitten, und müssen daher angepasst werden. Aufgrund der vielen Abhängigkeiten auf diese Klassen wird ein Refactoring benötigt.

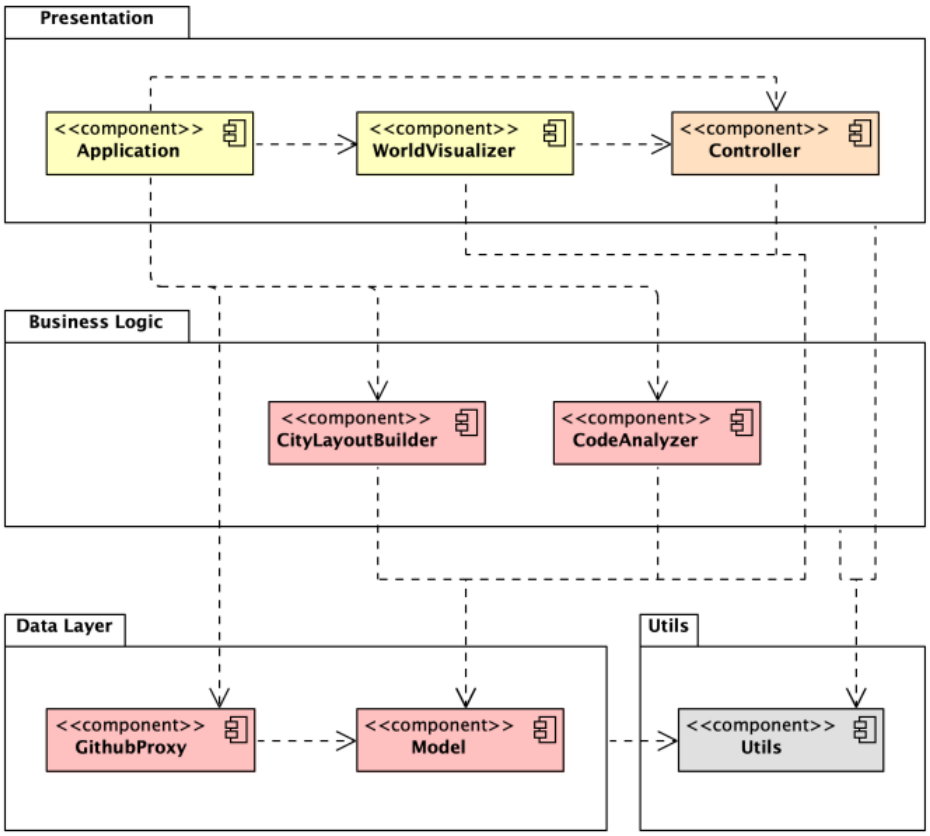

Abbildung 5: Komponentendiagramm vorher (Hindermann und Hirzel 2022a)

Die Abbildung 5 zeigt das Komponentendiagramm, zu Beginn des Projekts. Durch das Ersetzen der Codeanalyse, werden gewisse Komponenten nicht mehr gebraucht. Die Komponenten *CodeAnalyzer* und *GithubProxy* sollen komplett entfernt werden. Das Model soll an die neue Datenstruktur angepasst werden.

4.2 Komponentendiagramm

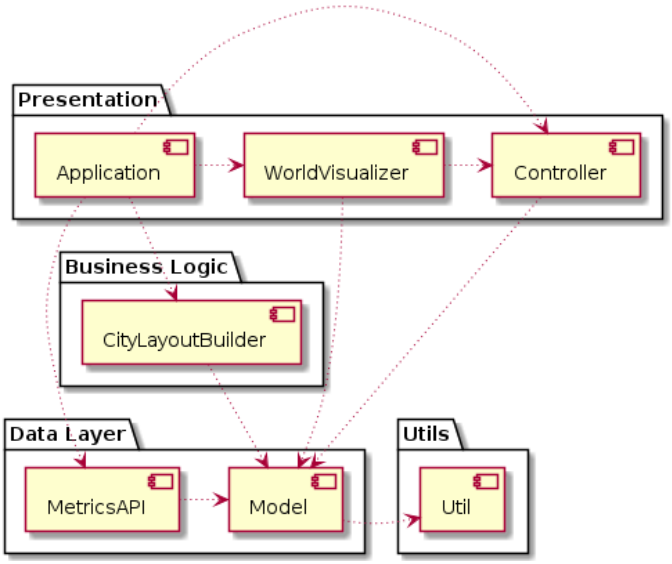

Abbildung 6: Komponentendiagramm nachher

Abbildung 6 stellt das überarbeitete Komponentendiagramm dar. Die CondeAnalyzer Komponente aus dem vorherigen Projekt, wird nicht mehr verwendet. Die GithubProxy Komponente wurde durch die MetricsAPI Komponente ersetzt, die für die Kommunikation mit dem Backend verantwortlich ist. Die Model Komponente wurde angepasst.

4.3 Ablauf

4.3.1 Kommunikation der einzelnen Softwarekomponenten

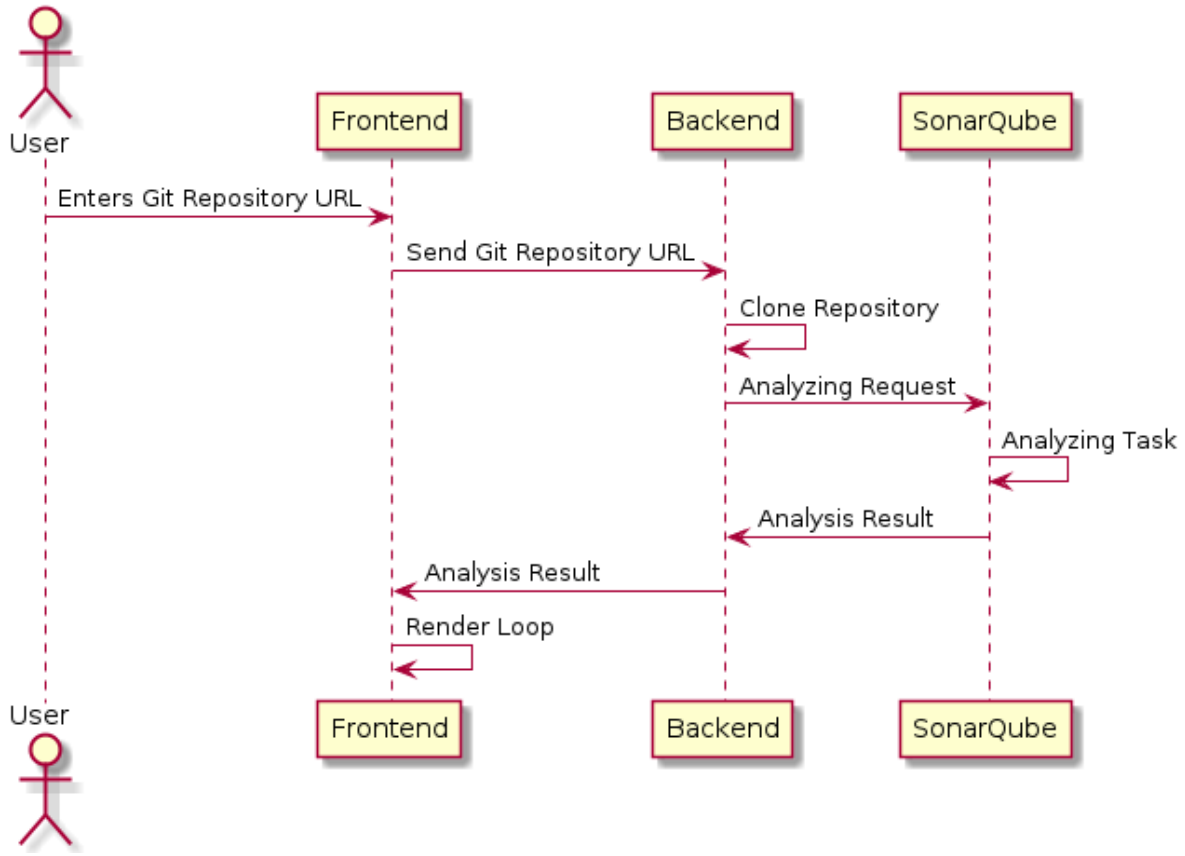

Abbildung 7: Sequenzübersicht

In Abbildung 7 ist die Kommunikation zwischen den einzelnen Softwarekomponenten dargestellt. Die einzelnen Komponenten werden im weiteren Verlauf dieses Dokumentes noch detaillierter dargestellt.

4.3.2 Frontend State Machine

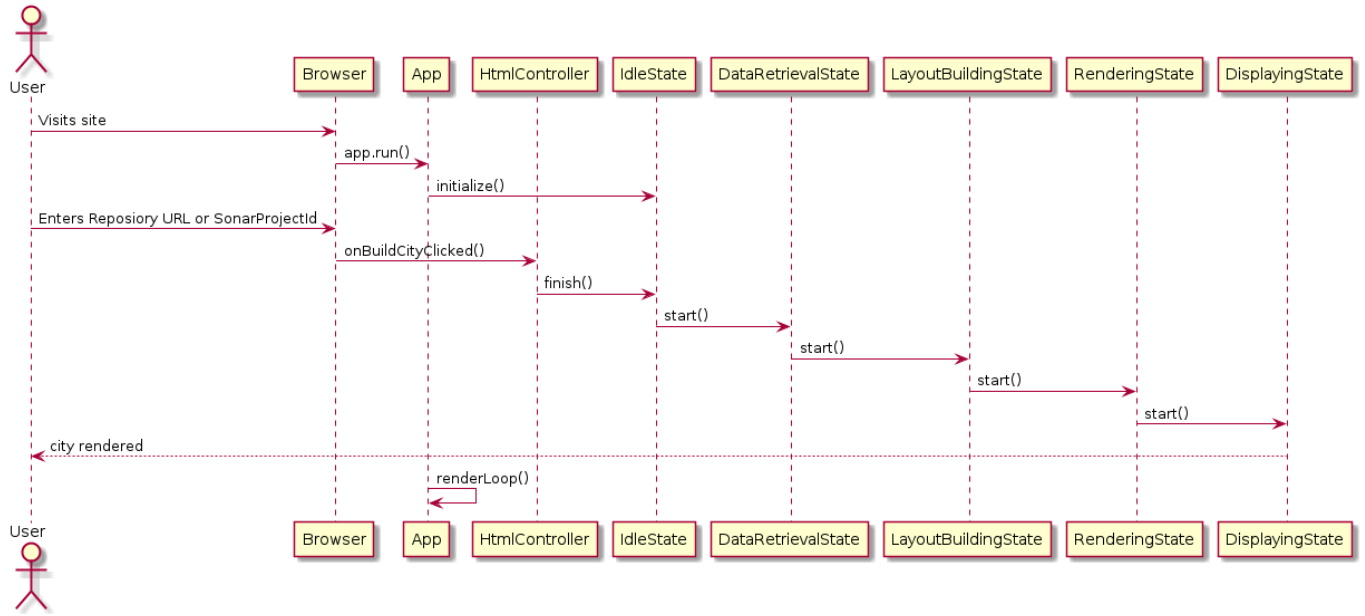

Abbildung 8: Sequenzdiagramm des Frontends

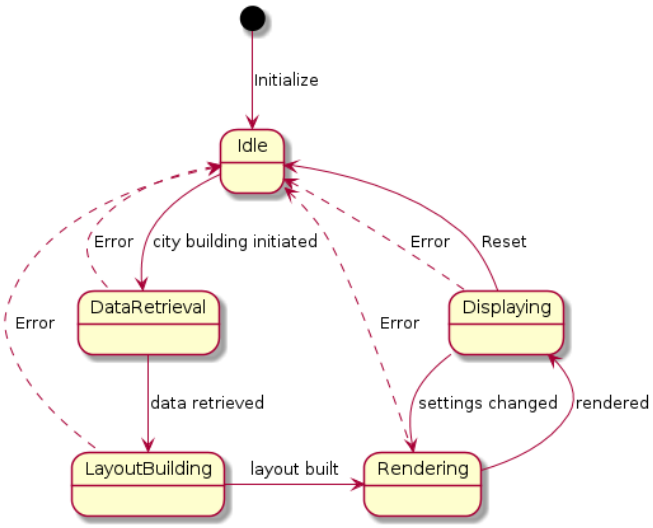

Abbildung 9: Zustandsdiagramm des Frontends

Die Abbildungen 8 und 9 zeigen den Ablauf und die Zustände grafisch dar. Beim Besuchen der Website wird die Applikation gestartet und auf den initialen Zustand (`IdleState`) gesetzt. Sobald Daten via Eingabefeld übermittelt werden, also entweder eine Repository URL, oder eine SonarQube Projekt ID angegeben und übertragen werden, wird das `onBuildCityClicked` Ereignis ausgelöst. Dieses Ereignis startet eine Kettenreaktion von mehreren Zustandswechseln. Jeder Zustand hat eine eigene Aufgabe, die direkt nach dem Erfolgen der Zustandsänderung gestartet wird. Bei erfolgreicher Beendigung der jeweiligen Aufgabe wird das Resultat an den nächsten Zustand übergeben und dieser anschliessend gestartet. Jeder Zustand kennt seinen Folgezustand. Falls in einem Zustand ein Fehler auftritt, springt die Applikation zurück in den initialen Zustand.

4.3.3 Data Retrieval State

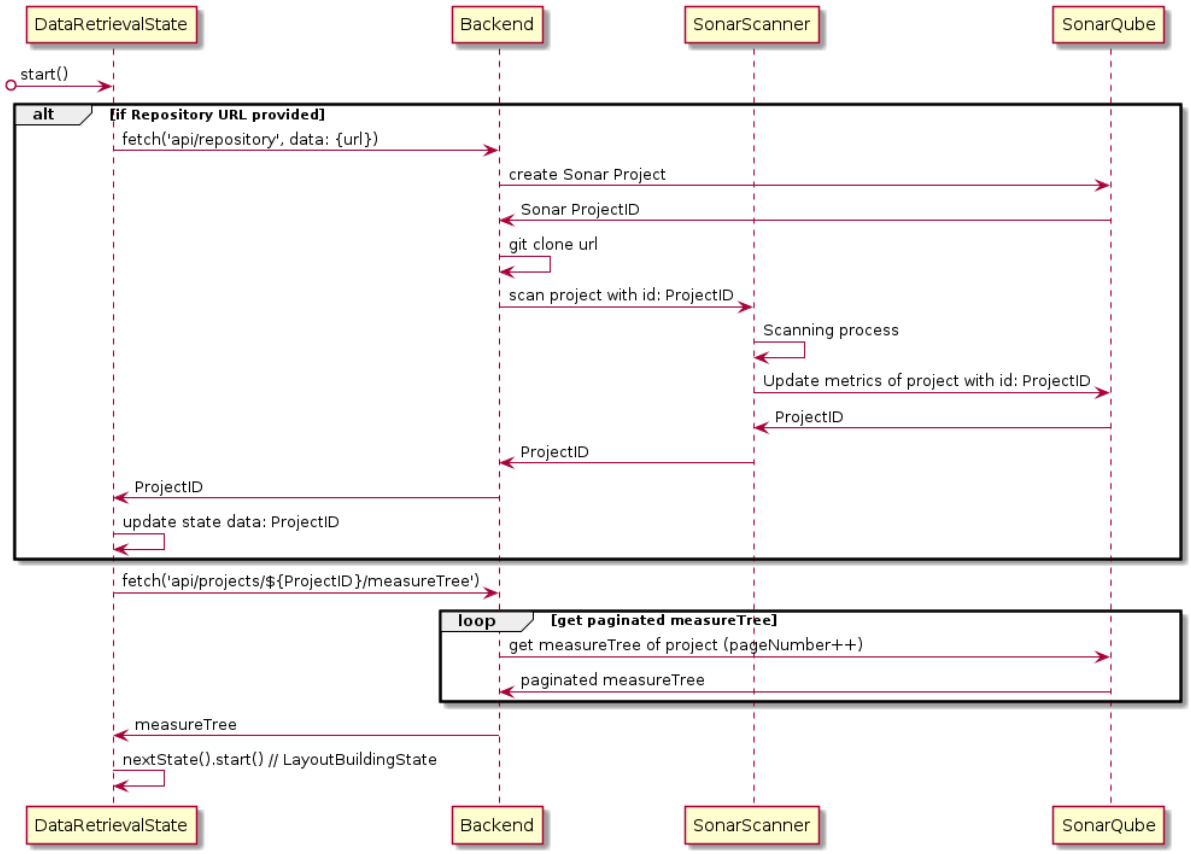

Abbildung 10: Sequenzdiagramm vom DataRetrievalState

Die Abbildung 10 stellt den bereits in Abbildung 8 erwähnten `DataRetrievalState` genauer dar. Bei der Eingabe einer Repository-URL wird ein neues Projekt in SonarQube erstellt, das Repository mit Git geklont und dieses schliesslich mit dem SonarScanner analysiert. Bei der Erstellung des Projekts in SonarQube wird eine ID generiert und an das Frontend übermittelt. Die ID kann verwendet werden, um den Analyseschritt in Zukunft zu überspringen. Von dem analysierten Projekt können nun die konkreten Werte der Metriken, die als Baumstruktur vorliegen (`measureTree`), von SonarQube über das Backend abgefragt werden. Diese Indirektion über das Backend wird eingebaut , da, je nach Grösse der Baumstruktur, nicht alle Daten mit einer Abfrage erhalten werden (Pagination). Das Backend enthält die Logik, um die ganze Baumstruktur abzufragen und anzureichern, damit sich das Frontend nicht darum kümmern muss.

4.3.4 Generierung und Darstellung der Stadt

Der Aufbau der Stadt wird mittels Squarified Treemap Algorithmus berechnet. Dieser unterteilt eine quadratische Fläche schrittweise in immer kleinere rechteckige Flächen. Dadurch eignet er sich für die Generierung einer Stadt, indem zuerst Stadtteile und Quertiere generiert werden und dann die Flächen für einzelne Gebäude. Die Flächen können dann mit Strassen und Wegen separiert werden. Der Algorithmus wird in dieser Arbeit nicht verändert. Eine ausführliche Erklärung ist in der Vorarbeit vorzufinden: (Hindermann und Hirzel 2022b)

4.4 Deployment

4.4.1 Entwicklung

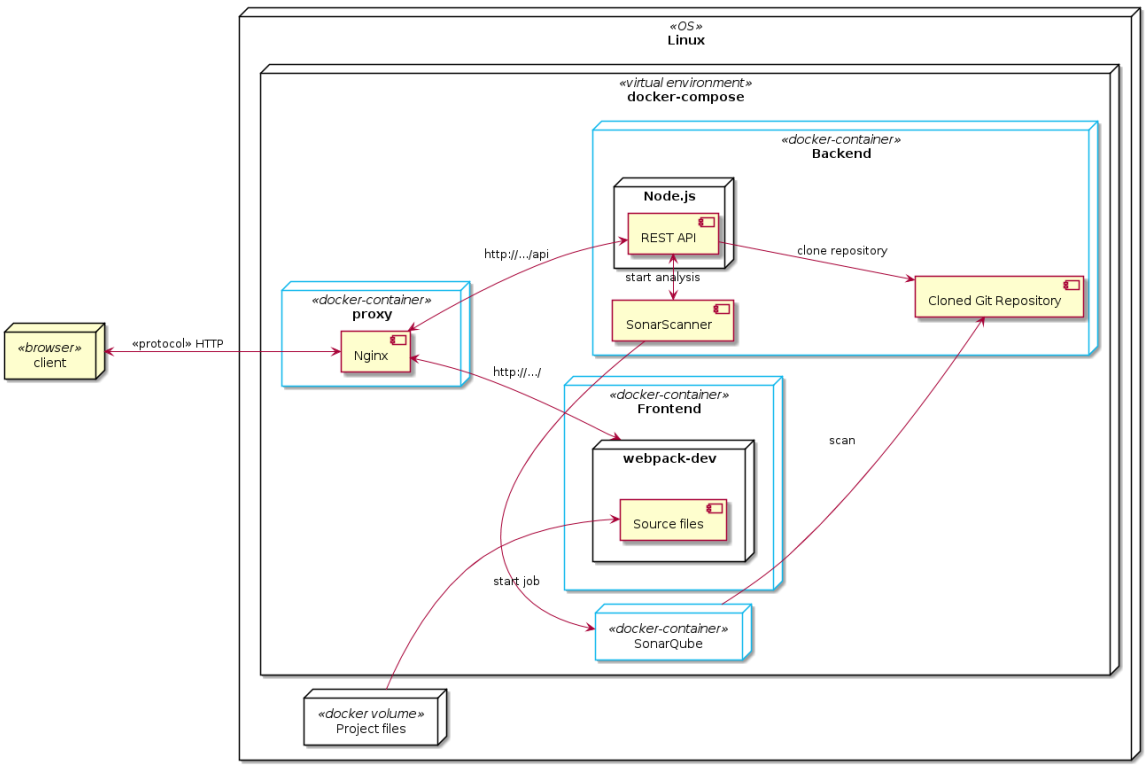

Abbildung 11: Deploymentdiagramm Entwicklung

Als Betriebssystem für die Entwicklung, wird aus Produktivitätsgründen Linux verwendet, sofern gerade keine VR-Brille fürs Testing benötigt wird. Falls eine VR-Brille verwendet werden soll, muss die Entwicklungsumgebung auf Windows ausgeführt werden, da die Oculus Software auf Linux nicht unterstützt wird. Docker-compose[1] wird für die virtuelle Umgebung verwendet, damit die ganze Infrastruktur, ohne zusätzliche Konfiguration und unabhängig des Betriebssystems, ausgeführt werden kann. Wie in Abbildung 11 ersichtlich ist, kann mittels Webbrowser auf den Nginx Proxy-Container zugegriffen werden. Dieser leitet alle Anfragen an den Frontend-Container weiter, ausser die angefragte URL entspricht der Form `http://.../api`, ist dies der Fall wird die Anfrage an den Backend-Container weitergeleitet. Der Frontend-Container führt einen Webpack[2] DevServer[3] aus, der den Quellcode bündelt und bei Änderungen am Quellcode die Webseite automatisch aktualisiert. Da es von Vorteil ist, den Quellcode ausserhalb der Dockerinfrastruktur bearbeiten zu können und die Dateien im Container immer auf dem aktuellsten Stand sein sollten, wird ein Dockervolumen verwendet. Der Backend-Container führt eine Node.js Applikation aus, die als REST-Schnittstelle fungiert. Diese Schnittstelle hat zur Aufgabe, Git-Repositories zu klonen, Analysen mittels SonarScanner zu starten, sowie Analyseresultate zu returnieren.

[1] Tool zur Definierung von Multi-container Applikationen: https://docs.docker.com/compose/
[2] OpenSource JavaScript bundler: https://webpack.js.org/
[3] Webpack Development Server: https://webpack.js.org/configuration/dev-server/

4.4.2 Produktion

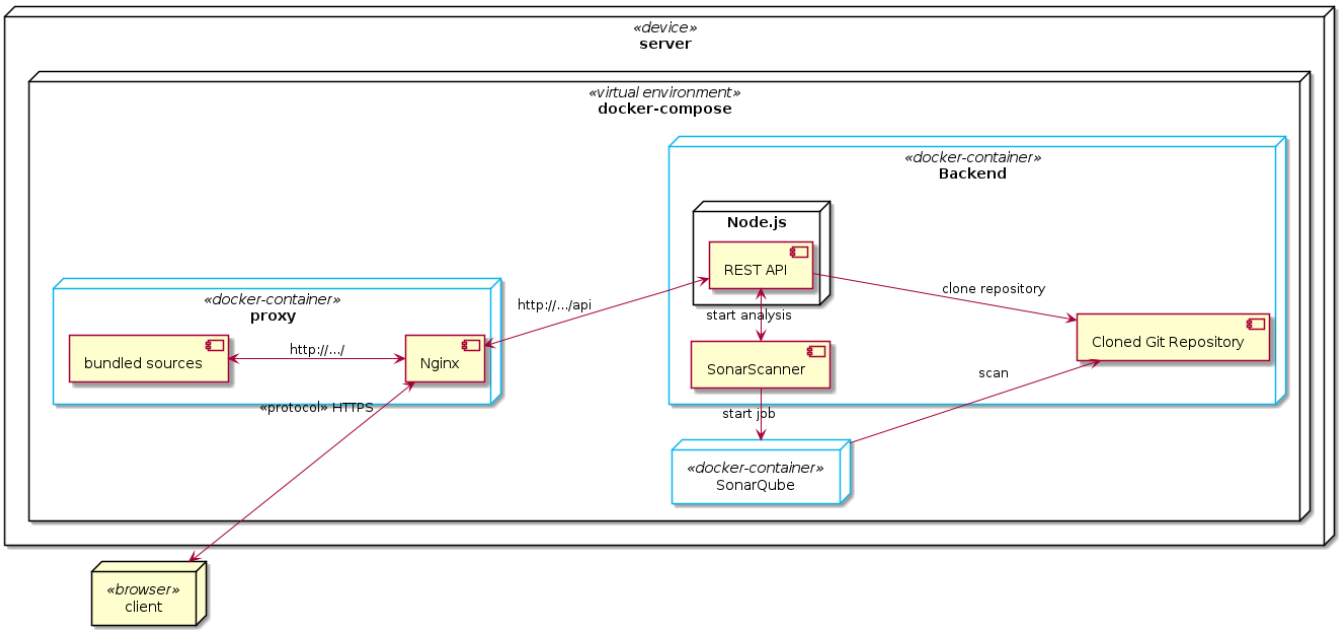

Abbildung 12: Deploymentdiagramm Produktion

In Abbildung 12 wird die Produktionsumgebung aufgezeigt. Im Unterschied zu der Entwicklungsumgebung werden die Quelldateien des Frontends vor dem Starten der Infrastruktur gebündelt und in den Proxy-Container kopiert. Denn in der produktiven Umgebung wird der Webpack[4] DevServer[5] nicht mehr benötigt.

[4] OpenSource JavaScript bundler: https://webpack.js.org/
[5] Webpack Development Server: https://webpack.js.org/configuration/dev-server/

5 Umsetzung

5.1 Werkzeug zur statischen Codeanalyse

Ein Werkzeug zur statischen Codeanalyse (SCAT) soll für die Messung von Metriken eingesetzt werden. Um das Passendste SCAT für unseren Anwendungsfall zu finden, werden die bekanntesten Werkzeuge evaluiert.

5.1.1 Kriterien zur Auswahl

Für die Evaluation der SCATs wurden folgende Kriterien bestimmt:

Sprachunterstützung: Software Cities soll mit den meist verwendeten Programmiersprachen kompatibel sein. Aus diesem Grund, ist es von Vorteil, wenn auch das SCAT möglichst viele Programmiersprachen unterstützt.

Metriken: Um die Qualität des Codes visuell darstellen zu können, sollen die Metriken möglichst vielfältig sein.

Schnittstelle: Das Werkzeug soll eine Anwendungsschnittstelle (API) besitzen, damit die gemessenen Metriken ausgelesen, und später für die Generierung verwendet werden können.

Freie Software: Es ist aus lizenztechnischen Gründen von Vorteil, wenn die verwendete Platform als freie Software verfügbar ist. Ein weiterer Vorteil, der sich ergibt, wenn ein Programm als freie Software angeboten wird ist, dass die Dokumentation ausgereift, und Schnittstellen nicht proprietär, also speziell für die Verwendung durch andere Programme, designt sind.

Hosting: Wenn das Werkzeug lokal ausgeführt werden kann, ergeben sich mehrere Vorteile. Einerseits werden dadurch die Zuverlässigkeit und Verfügbarkeit besser gewährleistet sein, andererseits ist eine bessere Kontrolle gegeben. So kann zum Beispiel die Version des Werkzeugs festgelegt werden, was bei Cloud-Anbietern meistens nicht möglich ist. Dadurch werden keine unerwarteten Änderungen an der API vorgenommen und die Kompatibilität ist gewährleistet.

5.1.2 Vergleich

Nach einer Recherche zu verschiedenen Werkzeugen wurden folgende Produkte herausgefiltert und genauer betrachtet. Aufgrund der Lesbarkeit, werden Funktionsumfang und Nicht-funktionale Eigenschaften auf zwei Tabellen aufgeteilt.

Tabelle 1: Vergleich des Funktionsumfangs der verschiedenen Werkzeuge

Produkt	Metriken	Sprachunterstützung	Schnittstelle
DueCode	Qualität, Schulden, Commit-Analysen	13 Sprachen	Nicht vorhanden
Snyk	Schwachstellen, Lizenzprobleme	8 Sprachen	Vorhanden
VeraCode	Schwachstellen	~ 15 Sprachen inkl. Plattformen & SDKs	Vorhanden
SonarCloud	Code-Duplikation, Testabdeckung, Schwachstellen, Bugs, Wartbarkeit, Zuverlässigkeit, Anzahl Zeilen, Kommentare	24 Sprachen	Vorhanden
SonarQube	Code-Duplikation, Testabdeckung, Schwachstellen, Bugs, Wartbarkeit, Zuverlässigkeit, Anzahl Zeilen, Kommentare	29 Sprachen	Vorhanden

Tabelle 2: Vergleich von nicht-funktionalen Eigenschaften der Werkzeuge

Produkt	Dokumentation	Lizenz	Hosting	Pricing
DueCode	Nicht vorhanden	Proprietär	Cloud	12$/Monat
Snyk	Gut dokumentiert	Proprietär	Cloud / Self Hosted (Docker)	Gratis (lowest Tier)
VeraCode	Gut dokumentiert	Proprietär	Cloud	Kostenpflichtig (Preise nicht öffentlich)
SonarCloud	Gut dokumentiert	Proprietär	Cloud	Gratis (lowest Tier)
SonarQube	Gut dokumentiert	LGPL v3	Self Hosted (Docker)	Gratis (Community Edition)

In Tabelle 1 ist zu erkennen, dass SonarQube und SonarCloud in Bezug auf Metriken das breiteste Spektrum abdecken. SonarQube weist mit 29 Sprachen die beste Sprachunterstützung auf, gefolgt von SonarCloud mit 24 Sprachen. Mit der Ausnahme von DueCode bieten alle untersuchten Werkzeuge eine Schnittstelle an, womit die ausgewerteten Metriken abgefragt werden können. In Tabelle 2 ist zu erkennen, dass nur zwei der aufgeführten Werkzeuge lokal ausgeführt werden können. Das einzige Produkt, das nicht unter einer proprietären Lizenz angeboten wird, ist SonarQube. Aufgrund dieser Eigenschaften wurde SonarQube für den Einsatz im Projekt ausgewählt.

5.1.3 SonarQube

SonarSource entwickelte SonarQube. Mit SonarQube erhalten Software Entwickler die Möglichkeit, Clean Code zu schreiben und zu pflegen, ohne dass dabei der Fokus auf die Arbeit verloren geht. SonarQube gibt es in verschiedenen Versionen, wobei die Community-Edition als freie Software veröffentlicht ist. Es können 29 Programmiersprachen analysiert werden. Um ein Softwareprojekt zu analysieren, wird ein SonarScanner eingesetzt. Es gibt verschiedene SonarScanner, die je nach Projekt-Build-Art (z.B. Maven, Gradle oder .NET) verwendet werden sollen. Nach dem Analyseprozess können via Weboberfläche oder Schnittstelle die Metriken eines Projekts abgefragt werden. Die Metriken werden im folgenden Kapitel erläutert.

5.2 Verwendete Metriken

SonarQube kann eine Vielzahl von Metriken ermitteln. Auf der offiziellen Webseite sind alle Metriken und deren Bedeutung aufgelistet. SonarQube verwendet vordefinierte Regeln, um Probleme bezüglich Zuverlässigkeit, Sicherheit oder Wartbarkeit zu erkennen. Die Regeln sind sprachabhängig und auf SonarSource öffentlich dokumentiert.

Gewisse SonarQube Metriken werden als eine Bewertung von A-E angegeben. Diese sind aus verschiedenen Einzelmetriken aggregiert, da sie alleine nicht feingranular genug sind, und somit häufig keine Aussagekraft besitzen.

Im Folgenden werden die Metriken erklärt, die von SonarQube abgefragt werden, um später das Modell der Stadt zu generieren.

5.2.1 Anzahl Codezeilen

Die Anzahl Zeilen, welche mindestens ein Zeichen enthalten, welches weder ein Leerzeichen noch ein Tabulatorzeichen oder Teil eines Kommentars ist.

5.2.2 Kognitive Komplexität

Die Zyklomatische Komplexität (Wallace, Watson, und Mccabe 1996) wird oft verwendet, um die Komplexität des Kontrollflusses einer Methode zu messen. Diese war ursprünglich dazu gedacht, Softwaremodule zu erkennen, die schwer zu testen oder warten sind und ist dafür geeignet die genaue Mindestanzahl von Testfällen zu berechnen, um eine Methode vollständig abzudecken. Sie ist aber kein zufriedenstellendes Mass, um die Verständlichkeit wiederzugeben. Als Abhilfe für dieses Problem wurde die kognitive Komplexität formuliert. Diese soll wiederspiegeln wie schwer es ist, den Kontrollfluss des Codes zu verstehen. (SonarSource S.A. 2021)

5.2.3 Duplizierte Zeilendichte (in Prozent)

SonarQube versteht unter dupliziertem Code mindestens 100 aufeinanderfolgende Zeichen, welche mehrfach vorkommen. Diese Zeichen müssen auf mindestens 10 Zeilen verteilt sein. SonarQube macht bei gewissen Programmiersprachen Abweichungen von diesen Regeln.

Die mathematische Definition der duplizierten Zeilendichte lautet wie folgt:

$$\frac{duplizierte\ Zeilen}{Zeilen} * 100$$

5.2.4 Wartbarkeitsbewertung

SonarQube kann auch den technischen Verschuldungsgrad bewerten. Die Definition dazu lautet:

$$\frac{Behebungskosten}{Kosten\,um\,eine\,Zeile\,zu\,entwickeln * Anzahl\,Codezeilen}$$

Die Behebungskosten werden berechnet, indem die geschätzten Behebungskosten aller “Code smells” aufsummiert werden. SonarQube rechnet in der obigen Formel damit, dass die Kosten um eine Zeile zu entwickeln 0.06 Tage betragen. Ein Ergebnis von fünf Prozent würde also bedeuten, dass fünf Prozent der Zeit, die bereits aufgewendet wurde, investiert werden muss, um die technischen Schulden abzubauen.

Mit dieser Kennzahl bewertet SonarQube zusätzlich die Wartbarkeit. Diese entspricht folgenden möglichen Werten:

- A: falls der Verschuldungsgrad kleiner oder gleich 5% ist
- B: Ein Verschuldungsgrad von 6 bis 10%
- C: Ein Verschuldungsgrad von 11-20%
- D: Ein Verschuldungsgrad von 21-50%
- E: Ein Verschuldungsgrad von mehr als 50%

5.2.5 Zuverlässigkeitsbewertung

SonarQube bewertet die Zuverlässigkeit der Applikation an der Anzahl und dem Schweregrad der gefundenen Bugs folgendermassen:

- A: 0 Bugs
- B: mindestens 1 kleiner Bug
- C: mindestens 1 grösserer Bug
- D: mindestens 1 kritischer Bug
- E: mindestens 1 blockierender Bug

5.2.6 Sicherheitsbewertung

SonarQube kann durch die vordefinierten Regeln auch Schwachstellen erkennen. Diese Regeln sind ebenfalls auf SonarSource öffentlich dokumentiert. Die Sicherheit der Applikation wird nach der Anzahl und dem Schweregrad der gefundenen Schwachstellen bewertet:

- A: 0 Schwachstellen
- B: mindestens 1 kleine Schwachstelle
- C: mindestens 1 grosse Schwachstelle
- D: mindestens 1 kritische Schwachstelle
- E: mindestens 1 blockierende Schwachstelle

5.2.7 Anzahl Security-Hotspots

Ein Security-Hotspot markiert einen sicherheitsrelevanten Teil des Codes, der von den Entwicklern überprüft werden muss. Bei der Überprüfung wird entweder festgestellt, dass keine Bedrohung besteht, oder es muss eine Kor-

rektur vorgenommen werden. Die Security-Hotspots können via Weboberfläche eingesehen werden und dort als “sicher” markiert werden, falls keine Bedrohung besteht. Im Unterschied zu den Schwachstellen besteht also die Möglichkeit eines “Falschalarmes”.

5.3 Metaphern

Das Ziel ist es nun, die ausgewerteten Metriken zu visualisieren. Dazu muss in einem ersten Schritt eine Zuordnung zwischen Metriken und Metaphern festgelegt werden. Die Metaphern sollen dabei möglichst einfach zu unterscheiden sein und in das Konzept der Softwarestadt passen.

5.3.1 Gebäude

Jede Datei des Projektes wird als Gebäude in der Stadt dargestellt. Dies entspricht dem bereits zuvor definierten Verständnis einer Softwarestadt.

5.3.2 Gebäudehöhe

Die Anzahl Codezeilen pro Klasse wird durch die Grösse und die Art des Gebäudes dargestellt. Eine grosse Klasse wird durch ein hohes Gebäude dargestellt, während eine kleine Klasse durch ein kleines Gebäude repräsentiert wird.

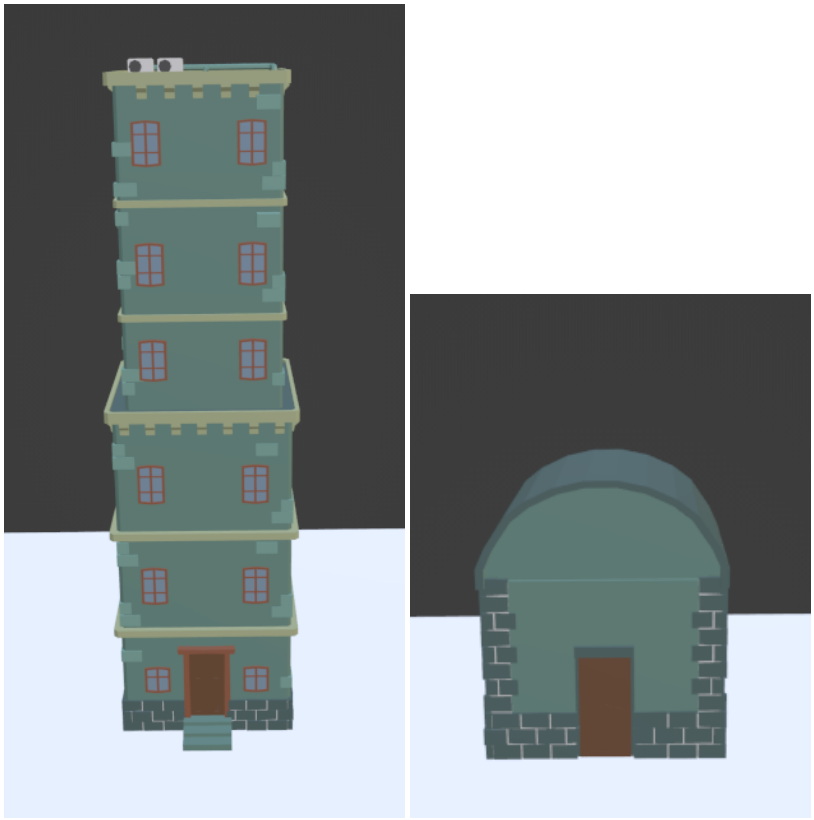

Abbildung 13: Links eine Klasse mit 200 Codezeilen und rechts eine Klasse mit 15 Codezeilen

5.3.3 Schräge Gebäude

Schwer zu verstehende Klassen, also Klassen mit einer hohen kognitiven Komplexität sollen als schräge Gebäude dargestellt werden.

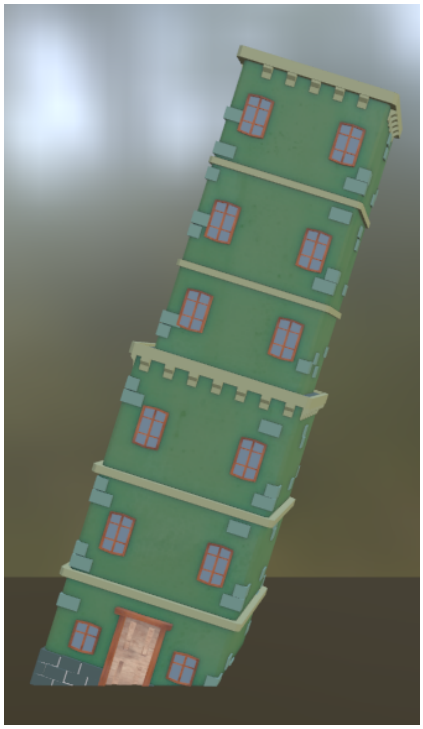

Abbildung 14: Eine Klasse mit einer hohen kognitiven Komplexität

5.3.4 Gebäudequalität

Wie bereits erwähnt, bewertet SonarQube die Zuverlässigkeit aufgrund der erkannten Bugs. Die Zuverlässigkeit soll durch die Qualität des Gebäudes, also das Modell, dargestellt werden. Falls die Zuverlässigkeit nicht mit “A” bewertet wird, soll ein beschädigtes Hausmodell dargestellt werden.

5.3.5 Farbe der Fassade

Die Farbe der Fassade kann mittels einer Albedo-Farbe in Babylon.js programmatisch verändert werden. Im Laufe des Projekts hat sich herausgestellt, dass die Farbe ein äusserst einfaches aber effektives Mittel darstellt, um Metriken zu visualisieren. Auch eignen sich Farben um mehrere Eigenschaften, in diesem Fall die Metriken, zu aggregieren.

Einerseits soll die Farbe der Hausfassade dazu verwendet werden, um die Wartbarkeit zu visualisieren. Aufgrund der Rückmeldungen durch die Nutzertests wurde ausserdem entschieden, dass zusätzlich auch die Zuverlässigkeit, die schon mittels Gebäudequalität dargestellt wird, miteinbezogen werden soll. So werden nun die Zuverlässigkeit und Wartbarkeit kombiniert, indem der schlechtere Wert dargestellt wird. Bei einer guten Bewertung wird die Fassade grünlich dargestellt, bei der schlechtesten Bewertung wird die Fassade rötlich angezeigt. Abbildung 15 illustriert alle verwendeten Farben.

Diese Farbmetapher der Wartbarkeit und Zuverlässigkeit wird nun mit der duplizierten Zeilendichte aggregiert . Die duplizierte Zeilendichte beeinflusst die Helligkeit der Farbe. Falls eine Klasse keinen duplizierten Code aufweist, bleibt die Helligkeit des entsprechenden Gebäudes unverändert. Im Falle einer hohen duplizierten Zeilendichte wird das Gebäude also ganz dunkel, schon fast schwarz dargestellt.

Im Folgenden ein vereinfachter Auszug des Codes, der die Farbe des Gebäudes berechnet.

```
const worseRating = Math.max(metrics.maintainabilityRating, metrics.
    reliabilityRating);
const color = ratingToColor(worseRating);
color *= percentageToFraction(duplicationDensity);
changeMaterialColorTo(rootMesh, BuildingMaterialName.Main, color);

function ratingToColor(rating: Rating) {
    switch (rating) {
        case 1:
            return Color3.FromHexString('#7aff6c');
        case 2:
            return Color3.FromHexString('#d9ff6b');
        case 3:
            return Color3.FromHexString('#fff46d');
        case 4:
            return Color3.FromHexString('#fcb172');
        case 5:
        default:
            return Color3.FromHexString('#ff7b75');
    }
}
```

Im ersten Schritt wird die schlechtere Bewertung der Wartbarkeit und Zuverlässigkeit ausgewählt, um damit die Farbe in der Funktion `ratingToColor` zu berechnen.

Abbildung 15: Farbcodes der Fassade

In Abbildung 15 werden die Farbcodes der `ratingToColor` Funktion dargestellt. Im zweiten Schritt wird die duplizierte Zeilendichte in eine Zahl zwischen in eine Bruchzahl verwandelt und mit der zuvor berechneten Farbe multipliziert. Damit wird die Helligkeit angepasst. Zuletzt wird die Farbe mittels `changeMaterialColorTo` auf die Gebäudefassade angewendet.

Abbildung 16: Links eine Klasse mit einer geringen Zuverlässigkeit, rechts eine Klasse mit einer positiv bewerteten Wartbarkeit und Zuverlässigkeit

Abbildung 17: Von links nach rechts: eine Zunahme der duplizierten Zeilendichte

5.3.6 Rauch und Feuer

Sicherheit ist ein wichtiger Aspekt in jedem Softwareprojekt. Deshalb soll eine auffallende und prägnante Metapher verwendet werden, um eine schlechte Sicherheitsbewertung durch SonarQube darzustellen. Es soll ein rauchendes Gebäude dargestellt werden, wenn die Sicherheit mit “B” bewertet wird, also mindestens eine kleine Schwachstelle besteht, oder maximal zwei Security-Hotspots gefunden wurden. Falls mehr als zwei Security-Hotspots existieren, oder die Sicherheit mit “C” oder schlechter bewertet wird, soll ein brennendes Gebäude dargestellt werden.

Abbildung 18: Links ein Gebäude mit Raucheffekt, rechts ein Gebäude mit Feuereffekt

5.4 Visualisierung

Die meisten 3D-Modelle, beispielsweise die Strassen und Fahrzeuge, wurden vom Vorprojekt übernommen und unverändert beibehalten. Die Gebäudemodelle wurden aufgrund neuer Anforderungen, nämlich dem Bedürfnis mehr Metriken präziser zu visualisieren, angepasst. Diese neuen Gebäudemodelle wurden nicht selbstständig modelliert, sondern von einer Webseite bezogen. Mehr Details dazu können im Anhang vorgefunden werden. Die Modelle wurden in das Projekt integriert und den Bedürfnissen angepasst. Die Modelle wurden texturiert und es wurden "beschädigte, ruinenartige" Versionen erstellt, die zur Differenzierung der Gebäudequalität benötigt werden. Zur Anpassung der 3D-Objekte wird Blender[6] verwendet. Bei der Anpassung der Objekte wurden Materialien und Texturen verändert.

5.4.1 Materialien

Materialien kontrollieren das Erscheinungsbild eines Objekts. Sie definieren unter anderem deren Farbe, Textur, die Interaktion mit Licht und deren Substanz.

Materialien bestehen aus drei Shadern:

- Surface Shader
  - Kontrolliert Texturen und die Interaktion des Lichts mit der Oberfläche des Objekts
- Volume Shader
  - Definiert das Aussehen des Inneren eines Objekts, nützlich bei teilweise transparenten Objekten, wie Feuer, oder Rauch
- Displacement Shader
  - Die Form der Oberfläche kann durch eine Verschiebung verändert werden. Dadurch kann die Oberfläche eines Objekts besser detailliert dargestellt werden

Texturen: Eine Textur wird vom Surface Shader verwendet, um die Farbe eines Punktes des Objekts zu berechnen. Die Farbe berechnet sich aus der Farbe der Textur, der Lage des jeweiligen Punktes des Objekts im Weltraum der Szene und verschiedenen Lichtquellen.

Alternative Texturen: In SoftwareCities werden verschiedene Texturen verwendet, um das Erscheinungsbild von Objekten zu verändern. Mit Babylon.js können Texturen von Objekten programmatisch verändert werden. Der Vorteil davon ist, dass für jedes Gebäude nur ein Objekt gezeichnet und importiert werden muss und zum Beispiel die Fassade im Nachhinein anders dargestellt werden kann.

Texturen können in Blender ganz einfach gezeichnet, und anschliessend exportiert werden, damit diese in Babylon.js verwendet werden können.

In Babylon.js lässt sich eine Textur eines Materials während des Imports des Objekts ersetzen:

[6] Freies and quelloffenes Softwareprogramm für 3D Modellierung, Animation und Rendering: https://www.blender.org

Listing 1: Beispielcode um die Textur eines Objekts zu verändern

```
BABYLON.SceneLoader.ImportMesh('', '../assets/', 'donut.glb', scene, function (
    newMeshes, particleSystems, skeletons) {
    newMeshes[0].scaling = new BABYLON.Vector3(20, 20, 20);
    scene.getMaterialByName('Material.003').albedoTexture = new BABYLON.Texture(
        '../assets/Donut_Texture2.png', scene);
});
```

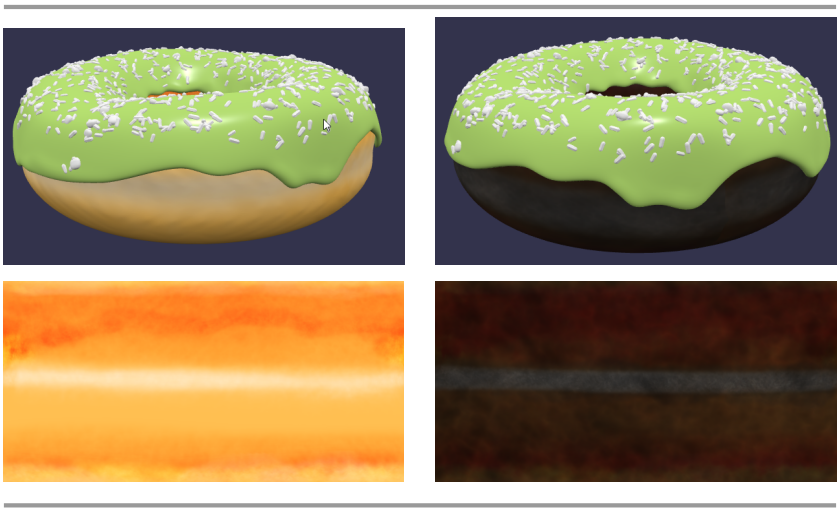

Abbildung 19: Links: Donut mit heller Textur, Rechts: Donut mit dunkler Textur

In Abbildung 19 sind zwei Donuts zu sehen. Diese Donuts wurden als Teil der Einarbeitung in Blender erstellt und dienten später als Testobjekte in Babylon.js. Die untere Spalte stellt die Textur des jeweiligen Donuts (ohne Glasur) dar. Mit dem Codeausschnitt in Abbildung 1 kann in Babylon.js die Textur eines Objekts, in diesem Fall eines Donuts, ganz einfach zur Laufzeit ausgewechselt werden.

6 Nutzerforschung

Es wurden Nutzertests mit zwei ausgewählten Nutzergruppen durchgeführt. In diesem Kapitel wird der Ablauf und das Resultat der Nutzertests dokumentiert. Des Weiteren werden die Einsatzzwecke der Software City untersucht und Verbesserungsvorschläge ausgearbeitet.

6.1 Ablauf der Nutzerforschung

Es wurde im Voraus ein Fragebogen definiert, der nach jedem Test durch die Testperson beantwortet wurde. Der Fragebogen ist im Anhang vorzufinden.

Im ersten Schritt wurden die Probanden zum Konzept und der Nutzung von Software Cities instruiert. Im zweiten Schritt wurde eines von zwei vorgegebenen Demoprojekten visualisiert, das von den Probanden mittels virtueller Realität erkundet wurde. Im nächsten Schritt wurde ein von den Probanden ausgewähltes Projekt visualisiert. Dieses konnte je nach Präferenz in VR, mit der *Oculus Quest 2*, oder im Browser dargestellt werden. Als Letztes musste die Stadt noch im Browser angeschaut werden, sofern dies nicht schon im vorhergehenden Schritt getan wurde.

Die Nutzertests wurden mit zwei Nutzergruppen durchgeführt:

- 3 Mitarbeitende der EPS Software-Engineering AG
- 2 Studierende des Bachelorstudiums an der Fachhochschule OST

6.2 Resultate

Im Folgenden werden die Resultate und Rückmeldungen zusammengefasst. Die originalen Antworten sind im Anhang vorzufinden.

6.2.1 Eindruck

Der allgemeine erste Eindruck der Testpersonen fiel positiv aus. Es wurde mehrfach erwähnt, dass das Interesse geweckt wurde und die Stadt neugierig erkundet wurde. Das Erlebnis wurde als “Eindrücklich” beschrieben. Die Visualisierung der Metriken wurde verstanden und ebenfalls positiv aufgefasst. Die abstrakte grafische Visualisierung wurde mehrheitlich als Vorteil angesehen gegenüber der Darstellung als “rohe” Statistiken, wie beispielsweise in SonarQube.

Bemängelt wurde, dass die Wiedergabe mit dem VR-Headset nicht flüssig war und dadurch das VR-Erlebnis beeinträchtigt wurde. Die Anzeige direkt im Browser wurde als flüssiger bezeichnet, aber auch nicht als ganz ruckelfrei. Die Tests zeigten auch, dass einige Benutzer zu Beginn Mühe mit der Bedienung der Menüs und der Applikation hatten. Dies ist möglicherweise auch darauf zurückzuführen, dass die Wenigsten zuvor in Kontakt mit virtueller Realität kamen.

Auf die Frage, ob die Stadt lieber mittels VR oder im Browser erkundet wurde, gab es keine eindeutige Antwort. Der VR-Modus wurde als eindrücklicher und spannender bezeichnet als die Bedienung der Weboberfläche und dadurch von gewissen Personen bevorzugt. Die Bedienung der Weboberfläche wurde aber als einfacher und nutzerfreundlicher bezeichnet.

6.2.2 Einsatzzwecke

Gewisse Einsatzzwecke wurden von verschiedenen Probanden vorgeschlagen. Zusätzlich wurde explizit nach einer Einschätzung in folgenden zwei Einsatzgebieten gefragt: der Einsatz in Sprint Retrospektiven und der Einsatz für Ausbildungszwecke. Im Folgenden sind mögliche Einsatzzwecke aufgeführt.

Scrumprozess und Code Reviews: Software City könnte im Scrumprozess in Sprint Retrospektiven Einsatz finden. Wenn es darum geht, Feedback des vorangegangenen Sprints zu erhalten, kann man die generierte Stadt von den jeweiligen Standpunkten im Team analysieren, und damit den Verlauf des Projekts anhand der Unterschiede in der Stadt erfassen. Des Weiteren könnte Software City, einem Product Owner, der häufig nicht mit dem Code vertraut ist, bei der Planung und Priorisierung des Sprint Backlogs behilflich sein.

Die Testpersonen wurden gebeten, auf einer Skala von 1 bis 10 zu bewerten, als wie sinnvoll der Einsatz von Software Cities in Sprint Retrospektiven eingeschätzt wird. Diese Frage wurde mit durchschnittlich 6.4 Punkten beantwortet. Es wurde mehrfach erwähnt, dass die Verwendung von Software Cities eine gute und sinnvolle Abwechslung in den Retrospektiven wären. Die eher tiefe Bewertung ist hauptsächlich auf zwei Kritikpunkte zurückzuführen. Einerseits wurde angemerkt, dass es nicht als nötig empfunden werde, die Stadt in jeder Retrospektive anzusehen. Andererseits wurde von einer Testperson ausgesagt, dass SonarQube selbst den Zweck auch erfülle. Manche Probanden vermuteten, dass durch das spielerische Element die Teilnehmer der Retrospektive dazu animiert werden aktiver teilzunehmen. Es wurde ausserdem suggeriert, dass der VR-Modus in einer Retrospektive weniger geeignet wäre, als die Webdarstellung auf einem grossen Bildschirm.

Ausbildung: Software Cities könnte einen Einsatzzweck in verschiedenen Ausbildungen finden. Studierende und Lernende mit geringen Programmierkenntnissen oder fehlender Projekterfahrung könnten von der Visualisierung möglicherweise profitieren. Der Lerneffekt wird möglicherweise gesteigert, wenn die Zusammenhänge zwischen der Stadtmetapher und dem Softwareprojekt erkannt werden.

Die Testpersonen wurden auch gebeten, auf einer Skala von 1 bis 10 zu bewerten, als wie sinnvoll der Einsatz von Software Cities für Ausbildungszwecke erachtet wird. Diese Frage wurde mit durchschnittlich 8.2 Punkten beantwortet. Ausserdem wurde die Frage gestellt, in welchem Abschnitt der Ausbildung der Einsatz am sinnvollen angesehen wird. Es wurde vorwiegend die Oberstufe (falls ein Informatikfach besucht werde) und die Lehre als Applikationsentwickler genannt. Es wurde von mehreren Probanden vermutet, dass die Aufmerksamkeit und das Interesse der Lernenden möglicherweise gefördert werde durch den Einsatz von Software Cities. Auch könne das Projekt eventuell zu einem besseren Verständnis beitragen, da eine neue Sicht auf Software geschaffen werde.

6.2.3 Konkrete Verbesserungsvorschläge und Ideen

Es wurden verschiedenste Ideen und Verbesserungsvorschläge von den Testpersonen genannt. Im Folgenden eine Auflistung davon:

- Performance verbessern
- Kontrast zwischen grünen Häusern und der Landschaft erhöhen (teilweise umgesetzt)
- Mehr Details auf dem Boden
- Unerwartet, dass Hausfarbe grün, auch wenn das Haus kaputt ist (umgesetzt)
- VR-Flugmodus zu schnell und hektisch (umgesetzt)
- Ein Modus, wo die Kamera automatisch die spannendsten Viertel und Gebäude der Stadt zeigt, um eine gute Übersicht über das ganze Projekt zu erhalten

Gewisse Punkte konnten gegen das Ende des Projektverlaufes noch gelöst werden. Erstens wurde die Fluggeschwindigkeit reduziert. Zweitens wurden die Farben der Hausfassaden angepasst, damit einerseits der Kontrast zwischen Landschaft und Häusern erhöht wird und andererseits die Häuser natürlicher aussehen. Schliesslich wurde noch die Bedeutung der Hausfarbe erweitert: anstatt nur die Wartbarkeit farblich darzustellen, wird nun die Wartbarkeit und Zuverlässigkeit dargestellt, wie in Kapitel Umsetzung beschrieben. Dadurch ist es nicht mehr möglich, dass eine Ruine grünlich dargestellt wird.

7 Zusammenfassung und Diskussion

In diesem Kapitel wird das Endergebnis des Projekts diskutiert und die Benutzersicht wird anhand von Bildern dargestellt.

7.1 Resultat

Das im Abstract definierte Ziel der Arbeit, also ein Softwareprodukt, das den Zustand eines Softwareprojekts mittels einer Stadtmetapher umsetzen kann, wurde ganz klar erreicht. Das Messen von Metriken und die Analyse von verschiedenen Programmiersprachen wurde ebenfalls erreicht. Es wurde Nutzerforschung betrieben, um herauszufinden, ob die Visualisierung von Softwareprojekten mithilfe von virtueller Realität, in einer realen Umgebung einen Einsatzzweck haben kann. Ausserdem wurden Erweiterungsmöglichkeiten für eine allfällige Folgearbeit überlegt.

7.2 Produkt

Das Resultat der Arbeit ist ein lauffähiges Produkt. Nachfolgend werden einige Screenshots gezeigt, die den Stand der momentanen Software aus Benutzersicht darstellt. Um dabei eine möglichst grosse Variation an Gebäuden zu erreichen, wurde ein Demoprojekt erstellt, das für die folgenden Screenshots als Basis verwendet wurde. Das Demoprojekt beinhaltet zum Teil bewusst komplexen und unsicheren Code.

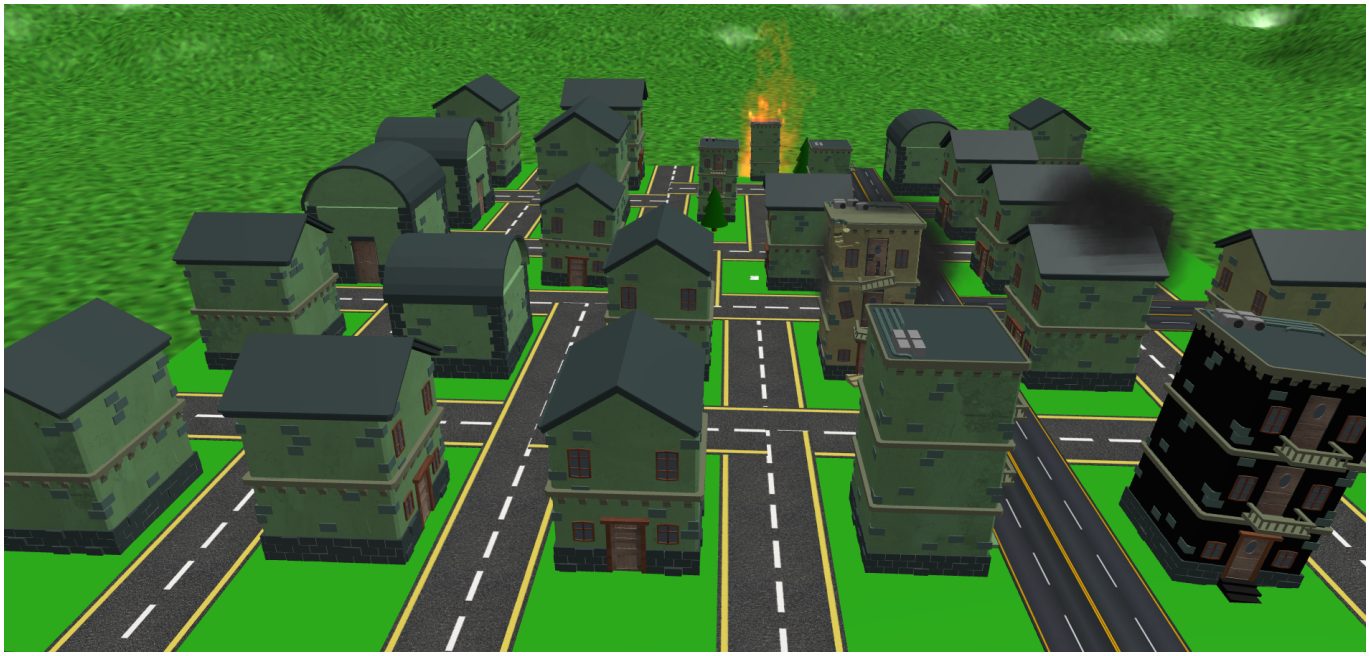

Abbildung 20: Ganze Stadt

In Abbildung 20 wird die gesamte Software Stadt des Demoprojekts dargestellt. Man kann Gebäude von unterschiedlichster Qualität darin erkennen.

Abbildung 21: Gebäude Textur schlecht

In Abbildung 21 können Gebäude mit verschiedenen Texturen erkannt werden. Die Qualität des Codes, der durch das linke Gebäude dargestellt wird, scheint bereits auf den ersten Blick, schlechter zu sein, als zum Besipiel der Code des rechten Gebäudes. Die Fassade ist kaputt, die Textur rötlich und Fenster sind eingeschlagen. Schlimmer wäre nur, wenn es auch noch brennen würde.

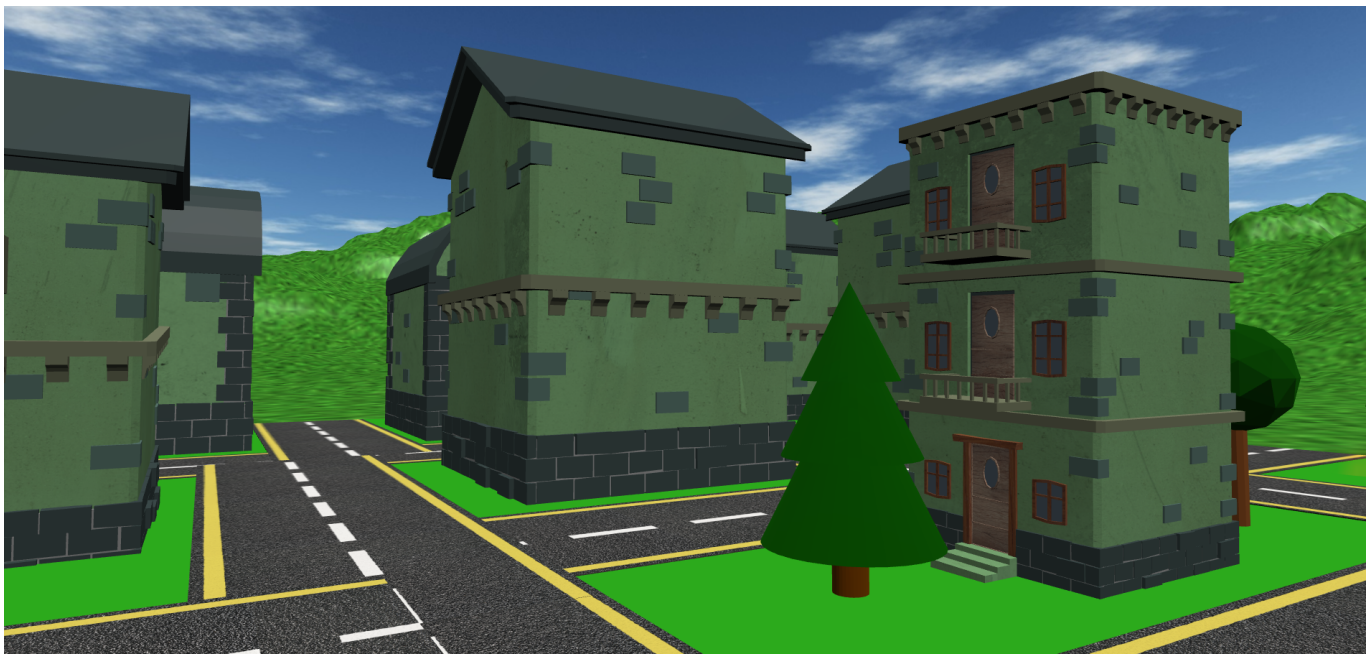

Abbildung 22: Gebäude Textur gut

In Abbildung 22 werden, im Gegensatz zum vorherigen Bild, Gebäude mit einer guten Qualität dargestellt.

Abbildung 23: Gebäude Textur brennend

Schlussendlich wird in Abbildung 23 noch das Worst Case Szenario dargestellt. Der Code dieses Gebäude beinhaltet offensichtlich Sicherheitslücken.

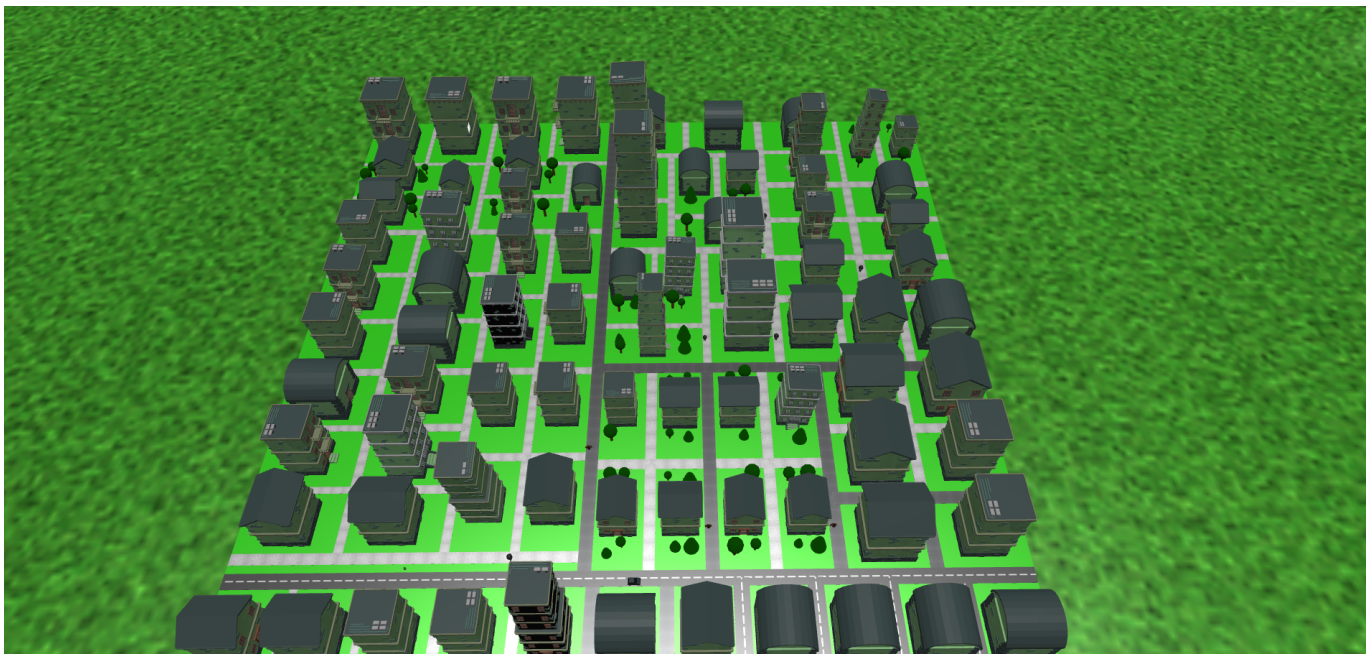

Abbildung 24: Grosse Stadt

Um einzelne Quartiere der Stadt zu veranschaulichen, wurde in Abbildung 24 ein etwas grösseres Projekt als Basis genommen. An den verschiedenen Breiten der Strassen können die einzelnen Quartiere erkannt werden.

7.3 Ausblick / Erweiterungsmöglichkeiten

Während dem Projektverlauf, dem Schreiben dieser Arbeit, und der durchgeführten Nutzerforschung sind auch einige Ideen für Erweiterungen entstanden.

7.3.1 Verwendung weiterer SonarScanner

Durch die Verwendung von SonarQube und SonarScanner, wurde die Unterstützung weiterer Programmiersprachen hinzugefügt. Zurzeit wird jedoch nur der standard SonarScanner verwendet. Um die tatsächliche Unterstützung für Gradle, Maven oder .NET Projekte zu gewährleisten, müssten weitere SonarScanner verwendet werden.

7.3.2 Zeitlicher Projektverlauf

Durch die Verwendung des Git Projekts für die Analyse, kann auch auf den Git Verlauf zugegriffen werden. Über den Verlauf könnte eine Animation der Stadt anhand des Projektverlaufs über Zeit visualisiert werden.

7.3.3 Weitere Anpassung der Metriken

Bei der Nutzerforschung wurde herausgefunden, dass die verwendeten Metaphern nicht immer intuitiv die Relation zur Qualität des jeweiligen Codes darstellen. Gewisse Metriken sollten demnach nochmals überarbeitet, oder aggregiert werden.

7.3.4 Ideen aus der Nutzerforschung

Bis zum Ende des Projekts konnten nicht alle Vorschläge aus der Nutzerforschung umgesetzt werden. Die unter Konkrete Verbesserungsvorschläge und Ideen genannten Änderungen sollten bei einer allfälligen Folgearbeit auch noch in Betracht gezogen werden.

8 Glossar

8.1 Erweiterte Realität (AR)

Erweiterte Realität, “augmented reality” oder AR. Ein Bereich des Realität-Virtualität-Kontinuums. Siehe: Abbildung 2 für die genaue Definition.

8.2 Freie Software und Open-Source Software

Software wird als “frei” oder “open-source” bezeichnet, wenn diese so lizenziert ist, dass jeder und jede die Möglichkeit hat die Software so zu benutzen, zu kopieren, zu untersuchen und zu verändern, wie er oder sie möchte. Aus historischen Gründen entwickelten sich die zwei unterschiedlichen Begriffe “frei” und “open-source”. Die beiden Begriffe werden heutzutage von den meisten Personen als Synonyme verwendet, auch wenn sich die Philosophien, welche sich hinter den Begriffen verstecken, teilweise unterscheiden. Die beiden Begriffe werden mit dem Überbegriff FOSS (free and open-source software) zusammengefasst. (Drake 2017)

Im Folgenden eine Definition von freier Software:

> Freie Software ist Software, die die Freiheit und Gemeinschaft der Nutzer respektiert. Ganz allgemein bedeutet das, dass Nutzer die Freiheit haben Software auszuführen, zu kopieren, zu verbreiten, zu untersuchen, zu ändern und zu verbessern. Ein Programm ist Freie Software, wenn Nutzer eines Programms über vier wesentliche Freiheiten verfügen: Die Freiheit, das Programm auszuführen wie man möchte, für jeden Zweck (Freiheit 0). Die Freiheit, die Funktionsweise des Programms zu untersuchen und eigenen Datenverarbeitungbedürfnissen anzupassen (Freiheit 1). Der Zugang zum Quellcode ist dafür Voraussetzung. Die Freiheit, das Programm zu redistribuieren und damit Mitmenschen zu helfen (Freiheit 2). Die Freiheit, das Programm zu verbessern und diese Verbesserungen der Öffentlichkeit freizugeben, damit die gesamte Gesellschaft davon profitiert (Freiheit 3). Der Zugang zum Quellcode ist dafür Voraussetzung. Freie Software wird dabei meist unter der GNU General Public License (GPL) lizenziert. (The Free Software Foundation 2021)

8.3 Freiheitsgrade

Die Anzahl Freiheitsgrade (“degrees of freedom” oder “DOF” auf Englisch) bestimmt wie viele Bewegungsmöglichkeiten ein Körper im Raum hat. Ein Freiheitsgrad kann als eine Koordinatenachse dargestellt werden. In Bezug zur virtuellen Realität wird meist von drei oder sechs Freiheitsgraden gesprochen. Mit drei Freiheitsgraden ist es möglich die Ausrichtung des VR-Headsets auszuwerten. Mit sechs Freiheitsgraden ist es zusätzlich möglich die Position des Headsets im Raum festzulegen.

8.4 Metrik

Eine Softwaremetrik widerspiegelt eine gewisse Eigenschaft von Quellcode und ist meist als mathematische Funktion definiert, die einen Zahlenwert abbildet. Beispiele von Softwaremetriken sind die Anzahl Codezeilen und die duplizierte Zeilendichte. Für weitere Informationen siehe: Messung von Codequalität.

8.5 Virtuelle Realität (VR)

Virtuelle Realität, “virtual reality” oder VR. Siehe: Definition virtueller Realität.

8.6 Web-XR

Web-XR ist eine Spezifikation, welche die Unterstützung für den Zugriff auf VR und AR Geräten, einschliesslich deren Sensoren und Displays, im Web definiert.(W3C 2022) Der Standard und die offene Schnittstelle sollen also ermöglichen, dass eine XR-Applikation unabhängig vom verwendeten Gerät, Browser und Betriebssystem benutzt werden kann.

8.7 Werkzeuge zur statischen Codeanalyse

Werkzeuge zur statischen Codeanalyse (auch “static code analysis tool” oder SCAT) werden dazu verwendet, um Quellcode zu analysieren, ohne diesen auszuführen. Der Quellcode wird analysiert, um Metriken zu erfassen und potenzielle Probleme, wie Qualitätsprobleme und Programmierfehler, zu erkennen.

9 Literaturverzeichnis

Bonasio, Alice. 2019. „Immersive experiences in education“. 2019. https://edudownloads.azureedge.net/msdownloads/MicrosoftEducation_Immersive_Experiences_Education_2019.pdf.

Drake, Mark. 2017. „The Difference Between Free and Open-Source Software“. Oktober 2017. https://www.digitalocean.com/community/tutorials/free-vs-open-source-software.

Goode, Lauren. 2019. „Get Ready to Hear a Lot More About 'XR'“. *Wired*, Januar. https://www.wired.com/story/what-is-xr/.

Graham, Hamish, Hong Yul Yang, und Rebecca Berrigan. 2004. „A Solar System Metaphor for 3D Visualisation of Object Oriented Software Metrics“. In *Proceedings of the 2004 Australasian Symposium on Information Visualisation - Volume 35*, 53–59. APVis '04. AUS: Australian Computer Society, Inc.

Hindermann, Thomas, und Joel Hirzel. 2022a. „Software Visualisierung durch VR mit Webtechnologien“. Januar 2022. http://sa_webvr.pages.gitlab.ost.ch/documentation/main.pdf.

———. 2022b. „Software Visualisierung durch VR mit Webtechnologien“. Januar 2022. http://sa_webvr.pages.gitlab.ost.ch/documentation/main.pdf.

„IEEE Standard for a Software Quality Metrics Methodology“. 1998. *IEEE Std 1061-1998*, i–. https://doi.org/10.1109/IEEESTD.1998.243394.

Knight, C., und M. Munro. 2000. „Virtual but visible software“. In *2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*, 198–205. https://doi.org/10.1109/IV.2000.859756.

Krasner, Herb. 2018. „The Cost of Poor Quality Software in the US: A 2018 Report“. September 2018. https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2018-report/The-Cost-of-Poor-Quality-Software-in-the-US-2018-Report.pdf.

Larman, Craig. 2004. „UML and Patterns“. September 2004. https://www.craiglarman.com/wiki/downloads/applying_uml/larman-ch6-applying-evolutionary-use-cases.pdf.

Milgram, Paul, Haruo Takemura, Akira Utsumi, und Fumio Kishino. 1994. „Augmented reality: A class of displays on the reality-virtuality continuum“. *Telemanipulator and Telepresence Technologies* 2351 (Januar). https://doi.org/10.1117/12.197321.

NASA Advanced Supercomputing. o. J. „Virtual Reality: Definition and Requirements“. https://www.nas.nasa.gov/Software/VWT/vr.html.

Nell, Lewis. 2020. „No smoke, no water, no waste. VR could train the next generation of firefighters“. 2020. https://edition.cnn.com/2020/01/29/tech/virtual-reality-firefighter-training/index.html.

Samadhiya, Durgesh, Su-Hua Wang, und Dengjie Chen. 2010. „Quality models: Role and value in software engineering“. In *2010 2nd International Conference on Software Technology and Engineering*, 1:V1-320-V1-324. https://doi.org/10.1109/ICSTE.2010.5608852.

SonarSource S.A. 2021. „Cognitive Complexity - a new way of measuring understandability“. April 2021. https://www.sonarsource.com/docs/CognitiveComplexity.pdf.

The Free Software Foundation. 2021. „What is Free Software?“ November 2021. https://www.gnu.org/philosophy/free-sw.html.en.

W3C. 2022. „WebXR Device API“. März 2022. https://www.w3.org/TR/2022/WD-webxr-20220314/.

Wallace, D, A Watson, und T Mccabe. 1996. „Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric“. Special Publication (NIST SP), National Institute of Standards; Technology, Gaithersburg, MD.

Wettel, Richard, und Michele Lanza. 2007. „Visualizing Software Systems as Cities“. In *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 92–99. https://doi.org/10.1109/VISSOF.2007.

4290706.

10 Anhang

10.1 Bedienungsanleitung

10.1.1 Vorbedingungen

- Node.js installiert
- Docker installiert
- Docker-compose installiert
- WebXR fähiger Webbrowser, zum Beispiel ein Chromium basierter Browser oder Firefox

10.1.2 Starten der lokalen Umgebung

```
## Zur Entwicklung im Projekt Root Ordner:
npm start
## oder
docker-compose up --build

## Um die Infrastruktur produktiv zu starten:
npm start:production
## oder
docker-compose -f docker-compose.prod.yml up --build
```

Mit den Befehlen wird mithilfe von Docker und docker-compose die ganze Infrastruktur (Frontend, Backend und reverse Proxy) installiert, konfiguriert und gestartet.

10.1.3 Verwenden der Software

Fall die Software lokal gestartet wurde, muss vor der ersten Verwendung SonarQube noch eingerichtet werden. Dazu muss über den Browser auf *localhost:9000* navigiert werden und sich mit dem Benutzer *admin* und dem Passwort *admin* eingeloggt werden. Anschliessend wird man aufgefordert, das Passwort zu ändern. Das Backend von InCode greift standardmässig mit dem Passwort *password* auf SonarQube zu.

Anschliessend kann über den Browser auf *localhost* InCode aufgerufen werden. Jetzt muss nur noch der Link eines öffentlichen Git Repositorys in der dazugehörigen TextBox eingegeben werden, und auf Start gedrückt werden.

Sobald das Backend mit der Analyse des Repositorys fertig ist, wird die Stadt direkt im Browser angezeigt.

Falls eine VR-Brille verwendet wird, muss nur noch auf den Enter-VR Knopf am oberen rechten Rand geklickt werden, dann wird die Stadt auf die verbundene VR-Brille übertragen.

10.2 Nutzerforschung

10.2.1 Fragebogen

Welches Projekt haben Sie sich mit Software Cities angeschaut?

Welche Rolle haben Sie in diesem Projekt?

Was war Ihr erster Eindruck von Software Cities?

Können Sie sich Use Cases vorstellen, bei denen Software Cities zum Einsatz kommen könnte, wenn ja, bei welchen?

Haben Sie die Stadt besser mit virtueller Realität erkunden können als im “Browsermodus”? Welchen Modus bevorzugen Sie?

Auf einer Skala von 1 bis 10: Als wie nützlich schätzen Sie den Einsatz von Software Cities für Sprint Retrospektiven ein? (Scrum Zeremonie) Begründen Sie.

Auf einer Skala von 1 bis 10: Wie sinnvoll schätzen Sie den Einsatz von Software Cities für Ausbildungszwecke ein? Begründen Sie.

Grundschule, Oberstufe, Lehrausbildung, Studium.

Haben Sie etwas Neues über das Softwareprojekt gelernt, welches visualisiert wurde?

Haben Sie noch allgemeine Anmerkungen / Anregungen?

10.2.2 Resultate

Teilnehmer 1 Welches Projekt haben Sie sich mit Software Cities angeschaut?

> *PTC & Demoprojekt 1*

Welche Rolle haben Sie in diesem Projekt?

> *Entwickler im PTC (firmeninternes Projekt)*

Was war Ihr erster Eindruck von Software Cities?

> *Interessiert, neugierig, was sind die Häuser, was kann ich in der Stadt machen / erkunden.*

Können Sie sich Use Cases vorstellen, bei denen Software Cities zum Einsatz kommen könnte, wenn ja, bei welchen?

> *Code reviews: schlechte Häuser fallen schnell auf Team reviews, Retros*

Haben Sie die Stadt besser mit virtueller Realität erkunden können, oder bevorzugen Sie den “Browsermodus”? Wieso?

> *Browsermodus: denn Performance besser & besser benutzbar: Files suchen, da keine Tastatur.*

Auf einer Skala von 1 bis 10: Als wie nützlich schätzen Sie den Einsatz von Software Cities für Sprint Retrospektiven ein? (Scrum Zeremonie) Begründen Sie.

> *6, ist brauchbar aber es erleichtert nichts Neues, Sonar als solches erfüllt Zweck ja auch.*

> *Aber gut, das spielerisches Element, aktiviert Leute aktiver mitzumachen in Retro.*

Auf einer Skala von 1 bis 10: Wie sinnvoll schätzen Sie den Einsatz von Software Cities für Ausbildungszwecke ein? Begründen Sie.

> *Zur Einführung in Programmierung / was ist Softwareprojekt: 6*

> *Am sinnvollsten in Oberstufe.*

Haben Sie etwas Neues über das Softwareprojekt gelernt, welches visualisiert wurde?

> *PTC: Ja, erstaunlich "grün", fast besser als erwartet.*

Teilnehmer 2 Welches Projekt haben Sie sich mit Software Cities angeschaut?

> *PTC & Demoprojekt 1*

Welche Rolle haben Sie in diesem Projekt?

> *PTC: Entwickler*

Was war Ihr erster Eindruck von Software Cities?

> *Gut, Stadt schaut interessant aus, vielfältige Häusermodelle. Performance leider nicht so gut in VR. Kontrast zwischen grünen Häusern und Hintergrund. Eventuell mehr Details auf Boden.*

Können Sie sich Use Cases vorstellen, bei denen Software Cities zum Einsatz kommen könnte, wenn ja, bei welchen?

> *Sprint Retro, privates Projekt überprüfen und anschauen. Schülern zu demonstrieren was ein Softwareprojekt ist, denn eindrücklicher als Stadt.*

Haben Sie die Stadt besser mit virtueller Realität erkunden können als im "Browsermodus"? Welchen Modus bevorzugen Sie?

> *VR ist spannender. Leider performance nicht so gut, aber VR bevorzugt.*

Auf einer Skala von 1 bis 10: Als wie nützlich schätzen Sie den Einsatz von Software Cities für Sprint Retrospektiven ein? (Scrum Zeremonie) Begründen Sie.

> *7, Nicht essenziell, aber spannend in Retro mal aufzuzeigen und zu visualisieren. Nicht in jeder Retro, aber zwischendurch / 1-2 Mal kann es spannend sein.*

Auf einer Skala von 1 bis 10: Wie sinnvoll schätzen Sie den Einsatz von Software Cities für Ausbildungszwecke ein? Begründen Sie.

> *8, Wäre sehr spannend, weil interaktiv und Schüler wären aufmerksamer und interessierter.*

> *Ab Oberstufe, falls Informatikfach mit programmieren. In Informatiklehre oder spezialisiertes (richtung Informatik) Studium auch sinnvoll.*

Haben Sie etwas Neues über das Softwareprojekt gelernt, welches visualisiert wurde?

> *Gelernt, dass die Tests viel duplizierten Code aufweisen. Sonst erwartungsgemäss, von Qualität, nichts sehr Unerwartetes.*

Haben Sie noch allgemeine Anmerkungen / Anregungen?

> *Farbe und Rauch / Feuereffekt könnte allgemein, also durchschnitt aller Metriken abbilden. Denn speziell, dass Haus kaputt aber trotzdem grün.*

Teilnehmerin 3 Welches Projekt haben Sie sich mit Software Cities angeschaut?

> *PTC & Demoprojekt 1*

Welche Rolle haben Sie in diesem Projekt?

> *Entwicklerin PTC*

Was war Ihr erster Eindruck von Software Cities?

> *Positiv überrascht, visualisierung der Gebäude gut. Gut gelöst, dass verschiedene Metriken verschieden dargestellt, direkt erkennbar welche Metriken schlecht, nicht einfach nur dass schlecht. Viel visueller und besser verständlich als "nur" mit Sonar.*

Können Sie sich Use Cases vorstellen, bei denen Software Cities zum Einsatz kommen könnte, wenn ja, bei welchen?

> *Schule, beim Lernen programmieren, aufmerksame Schüler, wenn spielerisch Stadt entdecken. Sprint retrospektive, dass spielerisch Projekt darstellen.*

Haben Sie die Stadt besser mit virtueller Realität erkunden können als im "Browsermodus"? Welchen Modus bevorzugen Sie?

> *Im Browser besser bedienbar, aber in VR mach es mehr Spass. In VR Flugmodus zu sensitiv.*

Auf einer Skala von 1 bis 10: Als wie nützlich schätzen Sie den Einsatz von Software Cities für Sprint Retrospektiven ein? (Scrum Zeremonie) Begründen Sie.

> *3, Nicht in jedem Sprint, aber ab und zu wäre es spannend.*

Auf einer Skala von 1 bis 10: Wie sinnvoll schätzen Sie den Einsatz von Software Cities für Ausbildungszwecke ein? Begründen Sie.

> *8, Schüler könnten dabei wahrscheinlich einiges lernen. Schüler wären aufmerksamer und gäben sich wahrscheinlich mehr Mühe, dass Gebäude nicht schwarz sein soll, als dass nur eine einfache Prozenzahl im Sonar dargesetellt werden würde.*

> *Oberstufe im Informatikfach und in der Lehre.*

Haben Sie etwas Neues über das Softwareprojekt gelernt, welches visualisiert wurde?

> *Erwartungsgemäss alles im grünen Bereich, ausser in den Tests einingen duplizierten Code, was aber bereits zuvor bekannt war.*

Teilnehmer 4 Welches Projekt haben Sie sich mit Software Cities angeschaut?

> *Demoprojekt 1 & Demoprojekt 2*

Welche Rolle haben Sie in diesem Projekt?

> *- (kein persönliches Projekt angeschaut)*

Was war Ihr erster Eindruck von Software Cities?

> *Eindrücklich, Landschaft, erstes Mal VR*

Können Sie sich Use Cases vorstellen, bei denen Software Cities zum Einsatz kommen könnte, wenn ja, bei welchen?

> *Bei Lehrling um zu veranschaulichen, wie wichtig Cleancode. Sollte eindrücklicher sein und Verständnis besser fördern. Im Büro auf Monitor, neben Build-status Stadtübersicht fürs ganze Team. Retro, in Routinearbeit einfliessen lassen.*

Haben Sie die Stadt besser mit virtueller Realität erkunden können als im “Browsermodus”? Welchen Modus bevorzugen Sie?

> *Im Browser besser bedienbar, Effekt aber besser und eindrücklicher in VR. (noch nie VR bedient)*

Auf einer Skala von 1 bis 10: Als wie nützlich schätzen Sie den Einsatz von Software Cities für Sprint Retrospektiven ein? (Scrum Zeremonie) Begründen Sie.

> *8: Ist sinnvoll, damit das Team sieht, wie das Projekt steht, und was noch Verbesserungspotenzial hat. Aber auch wichtig, dass keine gegenseitigen Anschuldigungen im Team dabei entstehen. Aber wichtig, dass Software Cities mit VR schnell aufgesetzt ist.*

Auf einer Skala von 1 bis 10: Wie sinnvoll schätzen Sie den Einsatz von Software Cities für Ausbildungszwecke ein? Begründen Sie.

> *9: Als Lehrlingsausbilder viel Cleancoderegeln mit Lehrlingen angeschaut. Der Einsatz von Software Cities wäre sehr sinnvoll um den Lehrlingen dies beizubringen und aufzuzeigen, wie sie im Projekt momentan stehen.*

Haben Sie etwas Neues über das Softwareprojekt gelernt, welches visualisiert wurde?

> *- (kein persönliches Projekt angeschaut)*

Haben Sie noch allgemeine Anmerkungen / Anregungen?

> *Flugmodus sehr hektisch, eventuell Geschwindigkeit reduzieren. Teleportier-modus aber angenehmer.*

Teilnehmer 5 Welches Projekt haben Sie sich mit Software Cities angeschaut?

> *Open source Projekt gtt-charts , Demoprojekt 2*

Welche Rolle haben Sie in diesem Projekt?

> *Entwickler und Maintainer von gtt-charts*

Was war Ihr erster Eindruck von Software Cities?

> *Sehr positiv, sieht Sinn dahinter um Vergleichsbasis zu haben und Code zu visualisieren. Gerade auch für unerfahrene Programmierer, eine "weniger abstrakte" Sicht auf den Code.*

Können Sie sich Use Cases vorstellen, bei denen Software Cities zum Einsatz kommen könnte, wenn ja, bei welchen?

> *Ja, in der Lehrlingsausbildung für Applikationsentwickler. Vorher-Nachher Vergleich um Qualität zu verbessern. Auch fürs Management könnte es einen sinnvollen Anwendungszweck darstellen.*

Haben Sie die Stadt besser mit virtueller Realität erkunden können als im "Browsermodus"? Welchen Modus bevorzugen Sie?

> *In VR sehr beeindruckend aber Browsermodus ist übersichtlicher. Browsermodus bevorzugt.*

Auf einer Skala von 1 bis 10: Als wie nützlich schätzen Sie den Einsatz von Software Cities für Sprint Retrospektiven ein? (Scrum Zeremonie) Begründen Sie.

> *8, weil eine gute High-level übersicht ohne trockene Metriken. Weniger trocken durch Gamification.*

Auf einer Skala von 1 bis 10: Wie sinnvoll schätzen Sie den Einsatz von Software Cities für Ausbildungszwecke ein? Begründen Sie.

> *10, siehe Begründung oben. Sinnvoll für alles bis ins frühe Studium.*

Haben Sie etwas Neues über das Softwareprojekt gelernt, welches visualisiert wurde?

> *War bei Visualisierung begeistert und positiv überrascht. Visualisierung war sehr solide (schöne grüne Häuser)*

Haben Sie noch allgemeine Anmerkungen / Anregungen?

> *Vorschlag: Um eine gute Übersicht über das ganze Projekt zu erhalten, wäre ein Modus super, wo die Kamera automatisch die spannendsten Viertel und Gebäude der Stadt zeigt.*

10.3 Externe Ressourcen

Viele der verwendeten externen Ressourcen wurden bereits im Vorgängerprojekt bezogen und werden in der Vorgängerarbeit ausführlich beschrieben: (Hindermann und Hirzel 2022a)

In dieser Arbeit wurden neue 3D-Modelle im Projekt hinzugefügt. Die Modelle wurden über Creazilla heruntergeladen und im Laufe des Projektes teilweise angepasst. Creazilla ist eine Webseite, die frei verfügbare Grafiken und Modelle bereitstellt, die von der Öffentlichkeit gratis verwendet werden können.

Folgende Modelle wurden in diesem Projekt verwendet: Buildings Pack 3d model | Creazilla. Die Modelle werden unter der creative commons 0 Lizenz zur Verfügung gestellt.

10.4 Projektplan

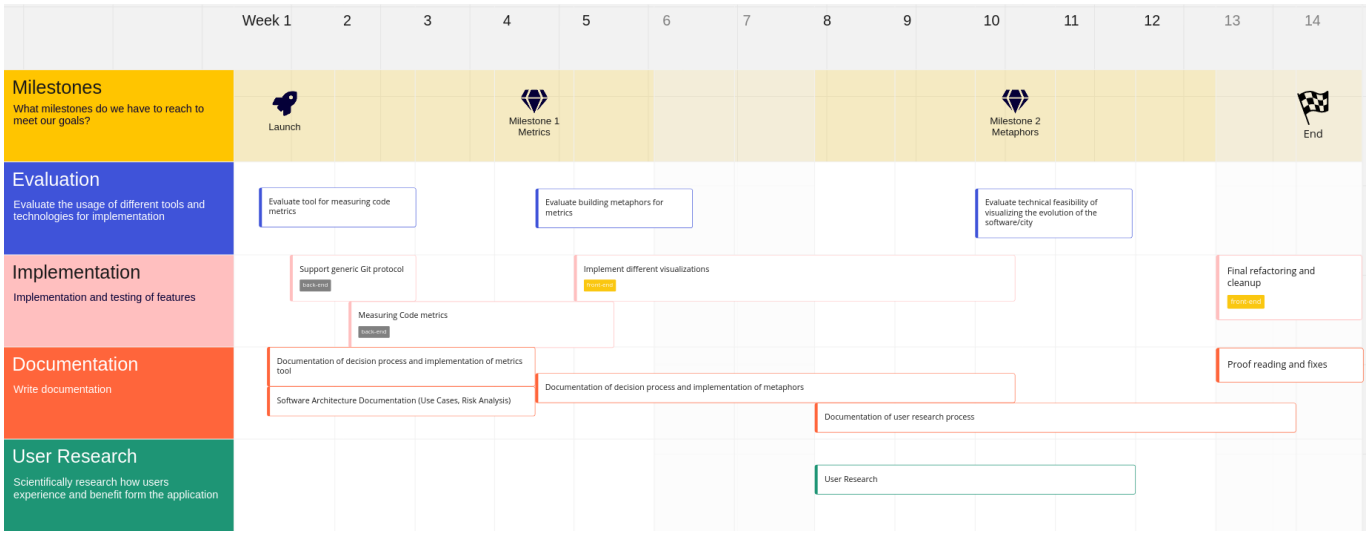

Abbildung 25: Projektplan

10.5 Aufgabenstellung

Entwicklung und Evaluation von immersiven und lebendigen Software Cities

Beteiligte Personen

Diese Arbeit wird verfasst von
Thomas Zahner; thomas.zahner@ost.ch
Benny Joe Villiger; benny.villiger@ost.ch

Betreuer dieser Arbeit ist
Prof. Dr.-Ing. Frieder Loch; frieder.loch@ost.ch

Problembeschrieb

In der Softwarevisualisierung werden Softwaresysteme visualisiert. Dies betrifft deren Struktur, Evaluation und die Darstellung von Metriken. Hierfür können Metaphern wie Software Cities eingesetzt werden. Diese stellen ein Softwaresystem in Form einer Stadt dar, bei der die Gebäude, zum Beispiel, die Module und deren Zusammensetzung repräsentieren. Dies soll eine niederschwellige und technologieunabhängige Möglichkeit schaffen, um einen Überblick über ein Softwaresystem zu gewinnen.

Die Verwendung von Webtechnologien für diese Anwendungsfälle verspricht offene und plattformübergreifende Lösungen. Dies steht im Gegensatz zur Verwendung von proprietären Game Engines, wie zum Beispiel Unity. Die Arbeit erweitert einen bestehenden Prototyp, der auf dem Framework babylon.js basiert und java-Repositories visualisieren kann. Ziel der Arbeit ist es, zu untersuchen wie Qualitätsmetriken in der Stadt visualisiert werden können. Ein Beispiel ist es, die Qualität einer Implementierung durch die Qualität der Fassade eines Gebäudes zu visualisieren.

Formulierung eines konkreten Auftrags

Die Arbeit identifiziert Qualitätsmetriken, die mit einer Softwarecity dargestellt werden können. Diese Metriken werden implementiert und es wird untersucht, ob diese dazu beitragen ein Verständnis über ein Softwaresystem zu gewinnen. Bei der Arbeit soll eine bestehende Anwendung weiterentwickelt werden.

Es sollen folgende Ergebnisse erarbeitet werden.

1. **Literaturrecherche** zu Softwarevisualisierung und Abgrenzung der Arbeit. Es werden wissenschaftliche Quellen und hochwertige Onlinequellen herangezogen und analysiert. Dies liefert die theoretische Grundlage der Arbeit und identifiziert die Forschungsfrage, die von der Arbeit adressiert wird.
2. Auf Grundlage der Literatur zur Softwarevisualisierung und den Möglichkeiten von Werkzeugen zur Qualitätsanalyse, zum Beispiel SonarQube, werden **darzustellende Metriken identifiziert** und beschrieben. Es wird die Einbindung dieses Werkzeuges in die Gesamtarchitektur evaluiert und beschrieben.
3. Es wird bestimmt mit welchen **Metaphern** diese in der Softwarecity visualisiert werden können. Dies betrifft zum Beispiel Codeduplikate oder Sicherheitslücken. Sicherheitslücken könnten, zum Beispiel, durch ein rauchendes Haus dargestellt werden.
4. Es wird geprüft wie die **Evolution** des Repositories und der Qualitätsmetriken visualisiert werden kann (History). Hierbei werden technische Rahmenbedingungen betrachtet.

5. Die Implementierung auf Basis der bestehenden Anwendung wird **geplant**. Die Architektur wird in Form eines Diagrammes dargestellt. Die Visualisierungen werden als Mockups entworfen. Die Qualität des Codes wird durch geeignete Werkzeuge sichergestellt.
6. Die umgesetzte Softwarecity wird **evaluiert**. Ziel ist es zu bestimmen, ob die Verwendung von Softwarecities zu einem besseren Verständnis eines Softwaresystems geeignet ist. Die Evaluation erfolgt mit nicht am Projekt beteiligten Personen und wird nach wissenschaftlichen Methoden geplant, durchgeführt und ausgewertet.
7. Es wird angestrebt die Ergebnisse auch in Form eines **wissenschaftlichen Artikels** (4–6 Seiten) zu publizieren.

Umfang und Form der erwarteten Resultate

Die Ergebnisse der Arbeit (Software und Dokumentation) sollen als Open Source-Projekt publiziert werden. So wird es ermöglicht, dass die Ergebnisse ohne Einschränkung von der OST und anderen Parteien weiterverwendet werden können. Bei der Veröffentlichung der Ergebnisse und den verwendeten Libraries werden Permissive-Licenses (z.B. MIT) bevorzugt.

Anfangs- und Abgabetermin

- Start der Bearbeitung: **21. Februar 2022**
- Abgabe der Arbeit: **3. Juni 2022 (17:00 Uhr)**

Zulässige Hilfsmittel und weitere Betreuung

Alle verwendeten Hilfsmittel werden in der Arbeit aufgeführt. Die Betreuung erfolgt durch die genannte Betreuungsperson. Es werden wöchentliche Besprechungen vereinbart.

10.6 Eigenständigkeitserklärung

Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: Rapperswil, 30.05.2022

Name, Unterschrift: Thomas Zahner, T. Zahner

Ort, Datum: Rapperswil, 30.05.2022

Name, Unterschrift: Benny Joe Villiger, Benny Joe Villiger

10.7 Einverständniserklärung zur Publikation auf eprints.ost.ch

Einverständniserklärung Publikation auf eprints.hsr.ch

☒ SA
○ BA

Titel der Arbeit: Lebendige Software Cities zur Visualisierung von Softwareprojekten

Team: Benny Joe Villiger, Thomas Zahner

Betreuer: Frieder Loch

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.
Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil, 30.05.2022

Name(n)	Unterschrift(en)
Thomas Zahner	T. Zahner
Benny Joe Villiger	Benny J. Villiger

10.8 Urherber und Nutzungsrechte

1. Vereinbarung

Ohne anderslautende Vereinbarungen stehen die Schutzrechte und das Know-how an der Studienarbeit oder Bachelorarbeit (nachfolgend ‚Arbeit' genannt) und an der in diesem Rahmen geschaffenen Güter, wie Software, sowohl dem Rechtsträger der OST Ostschweizer Fachhochschule, dem für die Arbeit verantwortlichen Professoren sowie dem Verfasser der Arbeit resp. Entwickler der in diesem Rahmen geschaffenen Güter, wie Software, zu.

Die genannten Parteien übertragen sich gegenseitig nicht exklusiv, jedoch unentgeltlich, weltweit, sachlich und zeitlich unbeschränkt die jeweiligen Schutzrechte und das Know-how an der Arbeit und an der in diesem Rahmen geschaffenen Güter, wie Software, einschliesslich dem Recht zur Weiterübertragung, ab. Entsprechend steht es jeder Partei zu, sämtliche Schutzrechte an der Arbeit resp. an der in diesem Rahmen geschaffenen Güter, wie Software, beliebig weltweit, zeitlich und sachlich unbeschränkt zu verwerten. Darunter fällt namentlich aber nicht abschliessend das Recht zur Lizenzierung in jeder Art, Umfang und Form, das Recht zur Bearbeitung und damit zur Nutzung z. B. der Software oder Komponenten hiervon als Grundlage eines neuen schutzfähigen Guts. Die Parteien erklären sich gegenseitig den Verzicht auf Namensnennung bei der Verwertung der Schutzrechte und des Know-how durch eine oder mehrere Parteien gemeinsam und stimmen namentlich zu, dass jede Partei allein unter ihrem eigenem Namen die Schutzrechte resp. das Know-how verwertet. Die vorliegende gegenseitige unentgeltliche Übertragung der Schutzrechte resp. des Know-how bezieht sich auch auf Verwertungsarten, welche heute noch nicht bekannt sind.

Rapperswil, den 30.05.2022 ..... T. Zahner
Student

Rapperswil, den 30.05.2022 ..... Benjamin Lüthi
Student

Rapperswil, den 30.05.2022 ..... Martin Tschudi Jochen
Betreuer der Studienarbeit

10.9 Zeiterfassung Dokumentation

Im Folgenden ein generierter Zeitreport des Dokumentationprojektes:

10.9.1 TIME STATS

- **total estimate**:
- **total spent**: 23d 3h
- **spent**: 23d 3h
- **thomas.zahner**: 12d 7h 50m
- **benny.villiger**: 10d 3h 10m

10.9.2 ISSUES

iid	title	spent	total estimate
44	Apply suggestions from Frieder - part 2	2d 20m	
43	List of figures	30m	
42	Update time report	20m	
41	Complete glossary	3h 20m	
40	Paperwork - sign and add forms	1h	
36	poster	4h	
35	Apply suggestions from Frieder	1d 45m	
34	Implementation - reliability & maintainability	2h	
33	Code quality - update section	2h	
32	Update SCAT comparison & fix table	2h 30m	
31	User research - part 2	7h 30m	
30	Kapitel - Resultate	7h	
29	Danksagung schreiben	30m	
28	Abstract publishen	3h 10m	
27	Aufgabenstellung hinzufuegen	30m	
26	User Research - User tests zusammenfassen	5h 30m	
25	User Research - Kapitel	1h	
24	Requirements - NFRs	10m	
20	Architektur - Anpassen und Beschreiben	1d	
19	Architektur - Struktur	1h 30m	

iid	title	spent	total estimate
18	Architecture - diagrams	1d 1h	
17	Begriffserklaerung - SoftwareCity beschreiben	3h	
16	Abstract schreiben	5h	
15	Visualisierung und verschiedene Arten erklären	7h 50m	
14	Begin user research chapter	4h 30m	
13	Requirements - Use Cases	6h	
12	Fixing location of Buildings inside city	6h	
11	Setup time reporting tool	3h	
10	Define use cases	30m	
9	Update glossary	2h 30m	
8	Endnotes instead of footnotes	3h	
7	Define metaphors	4h	
6	Restructure and clean up documentation	1d	
5	Meetings und Weiteres	3d 4h 35m	
4	Definition of VR and visualisation	1d 1h	
3	Roadmap erstellen	2h	
2	Automatically convert Markdown to PDF	1d 1h 30m	

10.10 Zeiterfassung Softwareentwicklung

Im Folgenden ein generierter Zeitreport des Softwareprojektes:

10.10.1 TIME STATS

- **total estimate**:
- **total spent**: 26d 5h 10m
- **spent**: 28d 4h 40m
- **thomas.zahner**: 13d 6h 40m
- **benny.villiger**: 14d 6h

10.10.2 ISSUES

iid	title	spent	total estimate
53	Refactoring	1d 4h 30m	
52	Automatically convert Markdown to PDF	1d 1h 30m	
51	Production configuration	3h 40m	
50	Quick cleanup	1h	
48	Fixing location of Buildings inside city	6h	
46	Adapt house colour behaviour	2h 30m	
45	Fix statistics menu page	6h	
44	Include security hotspots	3h 40m	
43	fix speed in vr	1h	
42	fix vr button location	2h	
41	Fix paging in backend	6h	
40	Add something on the floor of the houses	2h	
39	Metrics Message Box: Remove name from Quarter (Path)	1h	
38	Remove Threshold Settings	2h	
37	Update About Text in Settings Menu	1h	
36	Open Settings Menu Button	2h	
31	Fix Message Boxes in VR	1d 45m	
30	Fix statistic summary	7h 50m	
29	Apply texture to buildings	1d 6h 45m	
28	Prepare repo for demo and user research	3h	
27	Create and use new building models	2d 10m	
26	Fix CI pipeline	3h 45m	
25	Fix position of GUIs	3h	
24	Remove scaling of city	2h	
23	Render fire	2h	
22	Render smoke	3h 55m	
21	Display buildings	1d 7h	
20	Mockups erstellen	3h	
19	Fix pipeline	2h 30m	
18	Fix shared directory	1h 15m	
15	Remove indices from State classes	25m	
13	Get rid of Github Data	1d	

iid	title	spent	total estimate
12	Adapt MeasureTree to actual ProcessedComponentDto Tree	6h	
11	Improve Frontend	6h	
10	Research 3D models	2h 30m	
9	Blender & Babylon Intro	1d 7h	
8	Get metrics from SonarQube 2	3h 30m	
7	Get metrics from SonarQube	4h 30m	
6	Analyse repo programatically	6h	
5	Initialaufwand	1d 2h	
3	Website usable with and without VR headset	7h	
2	Research and choose static code analysis tool	3h	
1	Fetch repos via clone link	1d	

10.11 Wöchentliche Besprechungen

Es wurde jede Woche eine Besprechung mit Frieder Loch, dem Betreuer der Arbeit durchgeführt. Die Besprechungen wurden protokolliert und sind im Folgenden chronologisch aufgeführt.

10.11.1 Besprechung vom 22.02.2022

Besprechung der Aufgabenbeschreibung

- Evaluation eines Tools für die Identifizierung von Metriken
  - z.B. SonarQube
  - Codeduplikate, Sicherheitslücken, etc.
- Metaphern für Metriken
- Evolution des Repositories über Git History

Arbeit nächste Woche

- Einarbeitung in den Code
- Evaluierung Tool zur Identifizierung von Metriken

Weiteres

- VR-Brille Oculus Quest 2 ab nächster Woche in Bibliothek verfügbar
- Frieder hat auch eine Oculus Quest 2 bestellt.
- Es sollten Meeting Notes geschrieben werden

10.11.2 Besprechung vom 01.03.2022

Besprechung Besprechung der Änderungen betreffend der Architektur, also der einführung eines Backends.

Besprechung der Meilensteine und des groben Projektplanes.

Teilnahme am Fr. 04.03.22 an der Demonstration von InCode Gebäude 8, Raum in der Ecke zu oberst.

Arbeit nächste Woche

- Kommunikation mit Backend
- Dokumentation verfeinern
  - Statisches Analyse Tool
  - Literaturrecherche (Link zur Fachliteratur von Frieder)
    - Software Visualisierungen
    - Software City
    - Clean Code
    - Code Complete (2)
    - Software Craftmanship

10.11.3 Besprechung vom 08.03.2022

Arbeit nächste Woche

- Metaphern definieren
- Metriken aus Sonarqube
- gltf Format anschauen
  - Koennen z.B Texturen ausgetauscht werden?
- Lesen:
  - DatenVisualisierung

10.11.4 Besprechung vom 15.03.2022

Besprechung

- Demonstration Metriken Rest API
- Kurze (noch nicht abgeschlossene) Einarbeitung in Babylon & Blender besprochen

Arbeit nächste Woche

- Metaphern definieren
- Dokumentation weiterführen
- Dokumentation aufräumen, um im Laufe der Woche oder bei der nächsten Besprechung zu zeigen
- Einarbeitung Babylon & Blender

10.11.5 Besprechung vom 21.03.2022

Besprechung

Was haben wir gemacht

- Blender
- Babylon
  - Blender Objekte rendern
  - Texturen anpassen
- Refactoring von InCode Code
- Dokumentation
  - Aufgeräumt
  - Metriken und Metaphern ergänzt

Fragen

- Wie genderneutrale Sprache umsetzen?
  - Falls möglich vermeiden durch passive Formulierungen
  - Ansonsten keine bestimmte Vorgabe, solange einheitlich

Arbeit nächste Woche

- Stadt mit eigenen Daten rendern (Adapter?)
- Literaturverzeichnis am Ende des Dokuments(nicht Ende der Seite)
- Mockup / Skizze von geplanter Darstellung der Metriken

10.11.6 Besprechung vom 29.03.2022

Besprechung Demo der Software City, wo erstmals eine Sonar-Metrik (Anzahl Codezeilen) dargestellt wird.

Was haben wir gemacht

- Refactoring und Anpassung vom Frontend Code
- Erstellen eines Adapters, damit erste Sonar Metriken erfolgreich in der Stadt dargestellt werden
- Nach frei verwendbaren 3D Modellen gesucht. Idee: Creazilla verwenden
- Dokumentation
  - Endnoten anstatt Fussnoten
  - Glossar ergänzt

Fragen Müssen die 3D-Modelle von uns erstellt werden?

Nein, es dürfen auf jeden Fall existierende Modelle verwendet werden, solange es die Lizenz erlaubt.

Arbeit nächste Woche

- Mockup / Skizze von geplanter Darstellung der Metriken
- Weitere Metriken einbinden

10.11.7 Besprechung vom 05.04.2022

Besprechung

Was haben wir gemacht

- Mockups der Metriken erstellt und dokumentiert
  - Duplizierte Zeilendichte noch nicht umgesetzt, da schwierig, eventuell anders darstellen oder ganz weglassen?
- Weitere Anpassung vom Frontend Code
  - UI angepasst
  - CityLayoutBuilder angepasst, aber noch nicht fertig (technische Schwierigkeiten beim Verwenden von WebWorker in Unittests)
- Gitlab Pipeline geflickt, wobei gewisse Tests noch fehlschlagen

Fragen

Arbeit nächste Woche

- Anforderungen, use cases, etc. in Dokumentation aufnehmen
- CityLayoutBuilder fertigstellen
- Erste Metrik darstellen

10.11.8 Besprechung vom 12.04.2022

Besprechung

Was haben wir gemacht

- Refactoring des frontend Codes (noch nicht abgeschlossen)
- Darstellung der Metriken
  - Anzahl Zeilen Code in Klassen -> Grösse/Art der Gebäude
  - Kognitiv komplexe Klassen -> schräge Gebäude
  - Zuverlässigkeit der Klassen -> Farbe des Gebäude (nicht Art des Gebäudemodells da schwierig umzusetzen)
    - Zuverlässige Klasse -> grünes Gebäude
    - Unzuverlässige Klasse -> rotes Gebäude

Fragen

Arbeit nächste Woche

- Anforderungen und Use cases in Dokumentation aufnehmen
- Darstellung der Metriken verbessern
- Am Refactoring des frontend Codes weitermachen

10.11.9 Besprechung vom 19.04.2022

Besprechung

Was haben wir gemacht

- Refactoring des frontend Codes
- Zeiterfassung Statistiken aus GitLab holen und Report generieren mit GTT
- Use cases aufgeschreiben
- Darstellung der Zuverlässigkeit als Farbe (grün, gelb, orange, rot)
- Darstellung der Wartbarkeit als helligkeit der Farbe (intensive Farbe - schwarz)
- Rauch und Feuer rendern (noch nicht fertig)

Arbeit nächste Woche

- Qualmende / brennende Gebäude
- Verschiedene Gebäudemodelle, je nach Zuverlässigkeit & Wartbarkeit
- Garten / Wiese wenn genügend Zeit
- Dokumentation weiterführen
- User-research vorbereiten

10.11.10 Besprechung vom 26.04.2022

Besprechung

Was haben wir gemacht

- Use cases aufgeschrieben
- Rauch und Feuer rendern bei Klassen mit Sicherheitsproblemen
- Begonnen Häuser umzumodellieren zu Ruinen (noch nicht fertig)
- Anzeigen von den Ruinen bei schlechter Qualität

Arbeit nächste Woche

- Verschiedene Gebäudemodelle abschliessen
- User-research vorbereiten
- Dokumentation weiterführen

Arbeit an der Dokumentation Anpassung & Erweiterung der Dokumentation (nicht nur nächste Woche):

- Use cases erweitern: Team review, Sprint retrospective, etc.
- Weitere Literatur in Dokumentation
- Visualisierung: Was ist Visualisierung, wofür bracht man es?
- Verschiedene Arten der Visualisierung, Software-city als Unterkapitel: Visualisierung → VR → Software cities
- Anforderungsanalyse vor der Architektur
- Begriffserklärung: Virtuelle Realität, Immersion
- User research
- Zusammenfassung am Schluss

10.11.11 Besprechung vom 03.05.2022

Besprechung

Was haben wir gemacht

- Häuser ummodelliert zu Ruinen (Anzeigen der Ruinen bei schlechter Qualität)
- Begonnen Häuser zu texturieren
- Menüdialoge angepasst
  - Zusammenfassung der Statistiken angepasst (initiales Fenster)
  - Suche verbessert
  - Dialoge angepasst
- VR Verhalten verbessert
  - Häuser realistischer skaliert
  - Möglichkeit zwichen Flugmodus und Teleportiermodus zu wechseln

Fragen

- Begriffsdefinition der Visualisierung wie genau erweitern?
- Könnten Quellen zu Softwarequalität (Korrelation Qualität und Erfolg) und Clean code zur Verfügung gestellt werden?
  - Code complete
  - Clean code
  - Softwarequalität (Dirk Hoffmann)

Arbeit nächste Woche

- User research
- Erscheinungsbild Häuser (Texturen & Farben)
- Dokumentieren

10.11.12 Besprechung vom 10.05.2022

Besprechung

Was haben wir gemacht

- Häuser ummodelliert zu Ruinen
- Paging Unterstützung beim Holen der Sonar Metriken
- Häuser texturiert
- VR Verhalten verbessert
  - Fluggeschwindigkeit angepasst
  - Anzeige der Menüs optimiert
- Demoprojekt aufgesetzt, in welchem alle Metaphern ersichtlich sind
- User research durchgeführt
- Erscheinungsbild Häuser (Texturen & Farben)

Fragen

- Wo sollen wir mit der Dokumentation weitermachen? -> Abstract

Arbeit nächste Woche

- Abstract schreiben
- Dokumentieren

10.11.13 Besprechung vom 17.05.2022

Besprechung

Was haben wir gemacht

- User research (Teil 2) mit Mitstudenten durchgeführt
- Verwenden der Anzahl Security-hotspots als Metrik
- Farbe des Bodens verändert
- Dokumentation erweitert:
  - Mockups
  - Abstract & Management Summary
  - Architektur
  - Visualisierung & virtuelle Realität

Fragen

- Aufgabenstellung: haben wir die aktuellste Version?
- Aufgabenstellung: “prüfen wie Verlauf dargestellt werden kann” …
- Aufgabenstellung: Wissenschaftlicher Bericht, wie und was genau? (Resultate der Usertests?)
- Vorherige Arbeit zitieren: Beginn des Architekturkapitels so in Ordnung?

Aufgabenstellung in den Anhang einfügen.

Architekturdiagramme von vorheriger Arbeit übernehmen und zitieren.

Arbeit nächste Woche An Dokumentation weiterarbeiten:

- Nur bis tiefe 3 Titel nummerieren
- Bilder mit unterschrift und dann im Text referenzieren
- Tabellen mit überschrift

10.11.14 Besprechung vom 24.05.2022

Besprechung

Was haben wir gemacht

- Abstract

Fragen

- Wie Abstract korrigieren / absegnen?

Wurde von Frieder durchgelesen, ist in Ordnung, kann hochgeladen werden, wo es von Frieder noch bestätigt werden muss.

- Wie ist das Plakat zu gestalten?

Frieder hat uns als Beispiel das Plakat des Vorprojektes geschickt.

- Zip Archiv vom Code?

Ja, alles das mit der Arbeit zu tun hat zippen.

- Arbeit drucken?

Nein, ist nicht nötig.

Arbeit nächste Woche

- Korrigierter Abschnitt von Frieder verbessern
- Wie Software zu starten und bedienen ist (Vorbedingungen: node & npm installiert, …)
- Abstract Hochladen
- Kapitel 3,4,5 zu einem Kaptiel

- SonarQube beschreiben, wer es Entwickelt und wie alt…
- Kapitel Anforderungsanalyse vor Implementation
- Bei Einleitung der verwendeten Metriken: wieso Metirken teilweise aggregiert: nicht möglich dass so feingranular wie in Diagramm darstellen kann
- Kombinieren der Metriken mit den Mockups
- Modellierungsprozesse beschreiben & woher Modelle
- Auf Donutbild verweisen
- Ursprüngliches UML einfügen und Änderungen beschreiben, “Nachher-UML” nicht zwingend

10.11.15 Besprechung vom 31.05.2022

Besprechung

Was haben wir gemacht

- Software Implementation abgeschlossen
- Dokumentation verbessert
- Dokumentation fast abgeschlossen

Fragen

- Sind NFRs nötig zu beschreiben?

Arbeit nächste Woche

- Auf Poster: Bild mit Statistik ersetzen mit brennendem Haus
- NFRs: nicht nötig, Use-Cases als Header-Level 2
- Zu jedem Kapitel eine Zusammenfassung / einfache Einführung
- Unterschied Codequalität & Softwarequalität
- Messung von Codequalität in Einführung: Beispiele von Metriken & Verlinkung ins Glossar
- Introduction: Bilder von Software-City & Sonnensystem
- Use cases: schreiben, dass im Fully-dressed Format von Larmann & Buch zitieren
- Abstract anpassen, nach Verbesserungsvorschlägen