Visualisierung von OSGi-basierten Softwarearchitekturen - Eine vergleichende Evaluation der Usability in 2D und Virtual Reality

Masterarbeit

im Studiengang Computing in the Humanities
an der Fakultät Wirtschaftsinformatik und Angewandte Informatik
Otto-Friedrich-Universität Bamberg

vorgelegt von

Meike Barbara Anne Schaller

angefertigt am

Lehrstuhl für Medieninformatik
Universität Bamberg

in Kooperation mit

Deutsches Zentrum für Luft- und Raumfahrt
Simulations- und Softwaretechnik

Erstprüfer: Prof. Dr. Andreas Henrich
Abgabedatum: 29.03.2019

Abkürzungsverzeichnis

AR Augmented Reality

ASQ After Scenario Questionnaire

CDC Connected Device Configuration

DIN Deutsches Institut für Normung

DLR Deutsches Zentrum für Luft- und Raumfahrt

GOMS Goals, Operators, Methods, Selection Rules

GUI Graphical User Interface

IDE Integrated Development Environment

LOC Lines of Code

MCI Mensch-Computer-Interaktion

OSGi Open Source Gateway initiative

PDA Personal Digital Assistant

RCE Remote Component Environment

SUS System Usability Scale

UML Unified Modeling Language

VR Virtual Reality

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung der Arbeit

Die Entwicklung komplexer Softwarearchitekturen bringt neben all dem technologischen Fortschritt auch viele Herausforderungen mit sich: Source Code wird immer umfangreicher und somit unübersichtlicher, was die schnelle Auffassung eines Softwareprojekts erschweren kann. Um den größtmöglichen Erfolg sowohl auf Technologie– als auch auf Anwenderebene zu erzielen, ist es wichtig, die Software so darzustellen, dass die Komponenten und Abhängigkeiten eines Softwareprojekts vom Nutzer schnell erfasst und verstanden werden können. Der Bereich der Softwarevisualisierung hat sich genau dies zur Aufgabe gemacht, um es beispielsweise neuen Softwareentwicklern zu ermöglichen, sich schneller und effizienter in ein bestehendes Projekt einzuarbeiten.

In der vorliegenden Arbeit wird eine vergleichende Nutzerstudie zu zwei unterschiedlichen Softwarevisualisierungen durchgeführt, die beim Deutschen Zentrum für Luft- und Raumfahrt, kurz DLR, in Köln in der Einrichtung für Simulations- und Softwaretechnik entwickelt wurden: Eine Virtual Reality-Anwendung, basierend auf einer Inselmetapher, sowie das webbasierte Pendant in 2D, visualisiert durch einen kräftebasierten Graphen. Vorlage für beide Visualisierungen liefert die ebenfalls vom DLR entwickelte Open Source-Software *Remote Component Environment*[1], kurz RCE, welche das Zusammenführen von in unterschiedlichen Programmiersprachen geschriebenen Tools ermöglicht. Dadurch können Ingenieure und Wissenschaftler bei der Entwicklung und der Simulation komplexer Systeme unterstützt werden. Anwendung findet dies beispielsweise im Bereich des Flugzeugbaus, bei welchem Ingenieure und Entwickler aus unterschiedlichen Regionen und Fachbereichen gemeinsam den Entwurf eines Flugzeugs simulieren können [48].

Der aktuelle Stand der Forschung hat sich bisher vor allem mit Softwarevisualisierungen in 2D und 3D beschäftigt. Hierzu existiert eine Vielzahl an Studien. Forschungsarbeiten im Hinblick auf Vergleiche zwischen 2D und VR hingegen sind jedoch noch kaum vorhanden. Darüber hinaus mangelt es an Prototypen und Usability-Studien, die die Benutzerfreundlichkeit solcher Systeme umfassend evaluieren [19].

Ziel der innerhalb dieser Masterarbeit durchgeführten Nutzerstudie ist es, sowohl die Stärken als auch mögliche Schwachstellen in Bezug auf die Usability der beiden Visualisierungsformen aufzudecken. Darüber hinaus soll die Frage geklärt werden, ob eine Softwarevisualisierung in VR überhaupt hilfreich ist oder ob es den Nutzer eher überfordert.

[1] http://rcenvironment.de/

Dies soll durch den Einsatz zweier empirischer Methoden herausgefunden werden: Im ersten Schritt werden mithilfe eines Usability-Tests sogenannte *Task Scenarios* durchgeführt, bei denen die Probanden innerhalb der Software gewisse Aufgaben (Tasks) zu erfüllen haben. Während diese Methode vor allem objektive Metriken misst, soll mithilfe zweier standardisierter Fragebögen zusätzlich die subjektive Wahrnehmung der Nutzer erfasst werden.

Da sich RCE in ihrer Anwendung an Softwareentwickler richtet, sollen auch nur diese als Probanden an der Studie teilnehmen. Durch die Wahl geeigneter Hypothesen soll ein valides Untersuchungsdesign erzielt und die angeführten Forschungsfragen geklärt werden.

1.2 Aufbau der Arbeit

Der erste Teil der Arbeit erläutert die relevantesten Konzepte der Programmiersprache JAVA und führt in das Gebiet von modularer Software ein. Hier wird auch das OSGi Framework[2] und dessen Funktionsweise vorgestellt, welches die technische Grundlage für die beiden zu evaluierenden Visualisierungen bildet.

Das nachfolgende Kapitel führt in die Bereiche Usability und User Experience ein. Die wichtigsten Begriffe werden erläutert und voneinander abgegrenzt. Kognitionspsychologische Aspekte verdeutlichen die Notwendigkeit von Guidelines und Metaphern für die Gestaltung von Benutzerschnittstellen, was letztlich auch die Grundlage von Softwarevisualisierungen bildet. Auch eine Übersicht über die verschiedenen Methoden der Usability Evaluation findet sich in diesem Kapitel der Arbeit.

In Kapitel 4 wird das Gebiet der Softwarevisualisierung vorgestellt. Die Darstellung des aktuellen Forschungsstands sowie der verschiedenen Arten von Softwarevisualisierungen sollen ein umfassendes Bild über die Entwicklung dieses Forschungsgebietes der letzten Jahre vermitteln. Anschließend werden die beiden Visualisierungen, die in dieser Arbeit untersucht werden, vorgestellt.

Darauf aufbauend wird dann im fünften Kapitel der Hauptteil dieser Arbeit beschrieben: Die Planung und Durchführung der Usability-Studie. Ziele und Hypothesen der Studie werden formuliert sowie die verwendeten Methoden - Usability-Test und Fragebögen - erläutert.

Im Anschluss erfolgt die statistische Auswertung der quantitativ und qualitativ vorliegenden Daten mit der Präsentation der Ergebnisse, welche aus dem Usability-Test und den Fragebögen hervorgegangen sind. Um diese nachvollziehen zu können, wird zunächst die Wahl der statistischen Analyseverfahren erläutert. Die Interpretation der Ergebnisse schließt sowohl die innerhalb der beiden Visualisierungen durchgeführten Tasks als auch die Gesamtbewertungen über die beiden Systeme mit ein. Dadurch soll ein umfassendes Bild über die Stärken und Schwächen beider Anwendungen erzeugt werden.

Abschließend werden die aus dieser Arbeit hervorgegangenen wichtigsten Erkenntnisse diskutiert. Auch eine kritische Auseinandersetzung mit der Arbeit ist Teil dieses Kapitels. Zuletzt erfolgt ein Ausblick auf zukünftige Tendenzen im Bereich der Softwarevisualisierung und deren Evaluation.

[2] https://www.osgi.org/

2 Modulare Software

2.1 Grundlagen

Die Modularisierung von Software, also die Zerteilung eines Programms in mehrere, einzelne Komponenten, zählt heutzutage zu einer der fundamentalsten Methoden in der Entwicklung von komplexen und umfangreichen Softwaresystemen. Dieses Verfahren wird als *Divide and Conquer* (zu dt. *Teile und herrsche*) bezeichnet: Das System wird also nicht als ein einziger, monolithischer Block entworfen, sondern zunächst in kleinere Teilprobleme (Module) zerlegt, bis das Problem als lösbar (beherrschbar) bezeichnet werden kann. Ein Modul stellt somit eine abgeschlossene Einheit dar, die gewisse Funktionen bereitstellt. Es handelt sich beim Modularisieren also um einen Entscheidungsvorgang, bei dem die einzelnen Funktionen eines Programms auf die verschiedenen Module aufgeteilt werden. Darüber hinaus stellen Module über eine definierte Schnittstelle Services für andere Module bereit [51].

Die Vorteile sind neben der Reduzierung der Komplexität eines Programms die vereinfachte Weiterentwicklung von Softwaresystemen, da einzelne Module schnell erweitert beziehungsweise verändert oder ausgetauscht werden können. Die Aufteilung eines Programms in Module vereinfacht zudem die Arbeitsaufteilung, da mehrere Entwickler parallel an verschiedenen Modulen arbeiten können. Wichtig ist hierbei vor allem die saubere Schnittstellendefinition, die garantiert, dass die einzelnen Module später problemlos miteinander kompatibel sind. Zu erwähnen ist außerdem noch der Aspekt der Wiederverwendung, da sauber implementierte, modularisierte Funktionen schnell und einfach in andere Softwareprojekte importiert werden können [4].

Im Zusammenhang mit modularen Softwarearchitekturen ist es hilfreich, die Designprinzipien zu *Kopplung* und *Kohäsion* zu erwähnen. Bei der Entwicklung von modularer Software sollte immer eine lose Kopplung und eine hohe Kohäsion zwischen den Komponenten angestrebt werden [42]. Dabei beschreibt der Begriff der Kopplung, wie stark zwei Module miteinander zusammenhängen. Kohäsion hingegen präsentiert ein Maß dafür, wie stark die Abhängigkeit zwischen den Funktionen innerhalb eines Modules ist [39].

Daneben existieren weitere Prinzipien, die bei der Entwicklung von modularer Software berücksichtigt werden sollten. Da in dieser Arbeit aber der Fokus auf der Darstellung von Abhängigkeiten liegt, wird an dieser Stelle nicht weiter darauf eingegangen.

2.2 Java

Java als statisch typisierte, objektorientierte Programmiersprache wird vor allem in der Anwendungsentwicklung eingesetzt. Vor der Ausführung übersetzt der Compiler den Java-Quellcode in sogenannten *Bytecode*, wodurch er maschinenlesbar und somit plattformunabhängig gemacht wird. Um das Programm zur Laufzeit auszuführen, muss eine *Java Virtual Machine* (JVM) auf dem Zielsystem installiert sein.

Die Plattformunabhängigkeit des Bytecodes bedeutet, dass in Java geschriebene Programme auf unterschiedlicher Hardware und unterschiedlichen Betriebssystemen ausgeführt werden können, ohne dass Änderungen am Quellcode durchgeführt werden müssen. Auch eine erneute Kompilierung ist nicht nötig [28].

2.3 Die OSGi Service Platform

Bei der *OSGi Service Platform* handelt es sich um eine auf Java basierende Softwareplattform, die von der OSGi Alliance, einem Industriekonsortium, bestehend aus renommierten Unternehmen wie beispielsweise IBM, Oracle und Sun Microsystems, spezifiziert wurde. Durch OSGi wird die Integration und Steuerung verschiedener Softwarekomponenten ermöglicht, was für komplexe Softwareprojekte heutzutage zwar immens wichtig, von Java standardmäßig aber nicht unterstützt wird. Für diesen Zweck wurde die OSGi Service Platform entwickelt. Im Folgenden soll der Aufbau einer OSGi-basierten Softwarearchitektur skizziert werden.

2.3.1 Aufbau

Die Hauptkomponente der Service Platform ist das *OSGi Framework*, welches zur Erstellung und Ausführung dynamischer, modularer Systeme verwendet wird. Dabei verwaltet das Framework den Lebenszyklus von Modulen sowie deren Abhängigkeiten untereinander [29]. Darüber hinaus unterstützt es bei der Bereitstellung und Nutzung von Services, deren Funktion im folgenden Abschnitt noch näher betrachtet wird. OSGi setzt die Programmiersprache Java voraus. Java-Quellcode wird üblicherweise in Dateien mit der Endung `.java` gespeichert, welche auch als *Compilation Unit* bezeichnet werden, da diese Dateien dem Compiler als Eingabe dienen. Zusätzlich enthält jede Compilation Unit eine Typ-Deklaration, die eine gewisse Klasse definiert.

Die Modularisierung des Quellcodes erfolgt in Java durch die Unterteilung der Klassen in *Packages*. Das OSGi-Framework liefert hierfür einen *Container*, der dieses Konzept noch um sogenannte *Bundles* und *Services* erweitert und es ermöglicht, diese zur Laufzeit zu installieren, deinstallieren und zur Ausführung zu bringen [59]. Dies geschieht auf vier konzeptionellen Ebenen:

- Module Layer
- Service Layer
- Lifecycle Layer
- Security Layer

Listing 2.1 — Beispiel für eine Manifest-Datei eines Bundles aus dem RCE-Projekt.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: RCE Components DOE Common
Bundle-SymbolicName: de.rcenvironment.components.doe.common
Bundle-Version: 8.1.0.qualifier
Bundle-RequiredExecutionEnvironment: JavaSE-1.7
Bundle-Vendor: DLR
Export-Package: de.rcenvironment.components.doe.common
Import-Package: de.rcenvironment.core.datamodel.api,
    de.rcenvironment.core.utils.common,
    org.apache.commons.csv;version="1.1.0",
    org.apache.commons.logging;version="1.1.1",
    org.codehaus.jackson;version="1.9.13",
    org.codehaus.jackson.map;version="1.9.13",
    org.codehaus.jackson.node;version="1.9.13"
```

Dabei hat jede Ebene eigene Funktionen zu erfüllen. Das Module Layer stellt die Klassenerweiterung dar, in der die Bundles als zentrale Modularisierungseinheit definiert sowie deren Metadaten verarbeitet werden. Dabei werden sie in eigenen JAR-Archiven paketiert und verbreitet. Ein Bundle bildet eine abgeschlossene Einheit zusammengehöriger Klassen, deren Abhängigkeiten zu anderen Bundles oder Services eindeutig definiert ist. Informationen über ein Bundle werden in einer Manifest-Datei des dazugehörigen JAR-Archivs gespeichert (Listing 2.1). Das Service Layer stellt mit den Services Klasseninstanzen dar und steuert dadurch die Kommunikation zwischen den Bundles über Interfaces. Dies hat den Vorteil, dass so keine explizite Abhängigkeit zwischen den Bundles vorhanden sein muss. Ein Bundle kann dabei mehrere Service Components besitzen. Das Lifecycle Layer ermöglicht es, einzelne Module dynamisch in ein System zu laden, während das Security Layer als optionale Ebene diverse Sicherheitsaspekte berücksichtigt und die nötige Infrastrukur zur Bereitstellung und Kontrolle von Sicherheitsapplikationen liefert [35].

Neben den einzelnen Schichten nimmt der *Classloader* eine zusätzliche, wichtige Funktion ein. Jedes Bundle besitzt einen eigenen Classloader, der dafür sorgt, die Klassen eines Bundles zu laden. So können sich Klassen aus unterschiedlichen Bundles gegenseitig referenzieren. Das Laden der Klassen über den Classloader hat den Vorteil, dass die Klassen der einzelnen Bundles voneinander isoliert werden, womit ein Zugriffsschutz bewirkt wird [59].

Abbildung 2.1 verdeutlicht anschaulich, wie die einzelnen Komponenten innerhalb einer OSGi-Architektur miteinander interagieren. Damit die Bundles untereinander auf ihre Packages und die darin enthaltenen Klassen zugreifen können, müssen die Packages zunächst exportiert werden. Gleichzeitig muss das nutzende Bundle aber auch die Packages, auf die es zugreifen möchte, importieren [49].

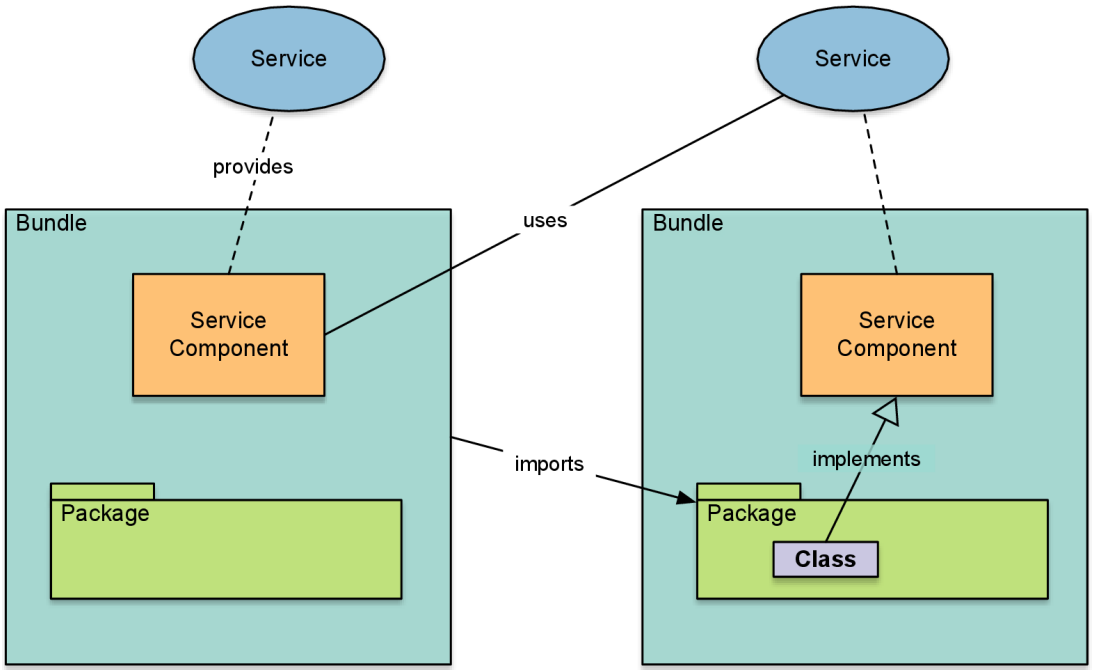


Abbildung 2.1 — Aufbau von OSGi [49].

Ein Bundle bietet Services an, die von anderen Bundles durch *Service Components* genutzt werden können [59]. Service Components werden durch XML-Dateien beschrieben und dienen der Deklaration von Services. Auch die in den Bundles enthaltenen Klassen können Services benutzen und müssen Service-Interfaces von bereitgestellten Services implementieren [28].

2.3.2 Beispiele

Es gibt diverse Implementierungen der OSGi-Spezifikation; zu den bekanntesten unter den Desktop-Applikationen zählen beispielsweise *Equinox*[1], welches die Grundlage der Eclipse IDE bildet. Andere Beispiele sind *Knopflerfish*[2] oder *Apache Felix*[3].
Daneben liegen weitere Einsatzgebiete in der Smartphone-Entwicklung: Auf *Connected Device Configuration*, kurz CDC, basierende Smartphones beispielsweise erlauben es, durch Nachinstallation neuer Komponenten diverse Funktionalitäten zu erweitern und anzupassen. Auch im Automobilbereich wird OSGi verwendet, hier überwiegend als Grundlage für Telematik- und Infotainmentsysteme [59].

[1] https://www.eclipse.org/equinox/
[2] https://www.knopflerfish.org/
[3] http://felix.apache.org/

3 Usability und User Experience

3.1 Abgrenzung der zentralen Begriffe

Software soll vom Anwender leicht und intuitiv zu bedienen sein. Aus diesem Grund ist das Usability Testing ein essentieller Bestandteil der Softwareentwicklung. Bei eingehender Betrachtung des Themas fällt auf, dass im Zusammenhang mit Usability auch andere Begriffe eine wichtige Rolle spielen, die im Folgenden näher beleuchtet und voneinander abgegrenzt werden sollen. Die Definitionen gründen dabei überwiegend auf den Spezifikationen des *Deutschen Instituts für Normung*, kurz DIN, welches zusammen mit der Europäischen Norm (EN) und der International Organization for Standardization (ISO) Standards zur Vereinheitlichung von Dienstleistungen und Gegenständen liefert.

Ergonomie

Der Begriff *Ergonomie* als Bezeichnung einer eigenen Wissenschaftsdisziplin für das Arbeitsumfeld existiert seit Mitte des 19. Jahrhunderts und wurde im Jahr 1959 von Brian Shackel erstmalig im Zusammenhang mit Computern erwähnt [50].
Laut DIN EN ISO 6385 [12], die die Grundsätze zur Gestaltung von Arbeitssystemen beschreibt, versteht man unter Ergonomie die Wechselwirkung aller Elemente, die man im Zusammenhang mit Arbeitssystemen findet. Dabei steht das Ziel im Mittelpunkt, das Wohlbefinden des Menschen und die Leistung des Gesamtsystems zu verbessern. Der Fokus liegt also nicht auf einem technischen System, sondern alle an einem Gesamtsystem beteiligten Bereiche, bestehend aus Mensch, Werkzeug, Aufgabe und Umgebung [46].

Usability

Mit der DIN EN ISO 9241, die 1997 eingeführt wurde, existiert eine internationale Normenreihe zur Beschreibung von Richtlinien der Mensch-Computer-Interaktion. Sie definiert Usability

> "als das Ausmaß [...], in dem ein technisches System durch bestimmte Benutzer in einem bestimmten Nutzungskontext verwendet werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen" [13].

Um die Usability eines Systems zu bewerten, muss also immer der Rahmen, in dem das System verwendet wird, berücksichtigt werden. Dabei kommt den Begriffen *Prozessangemessenheit* und *Aufgabenangemessenheit* größere Bedeutung zu, da technische Systeme immer in Abläufe oder Prozesse eingebettet sind und dazu dienen, innerhalb dieser Prozesse gewisse Aufgaben angemessen - also effektiv, effizient und zufriedenstellend - lösen zu können [46]. Die Effektivität beschreibt dabei die „Genauigkeit und Vollständigkeit, mit der Benutzer ein bestimmtes Ziel erreichen". Effizienz wird erklärt als der „im Verhältnis zu Genauigkeit und Vollständigkeit eingesetzte[r] Aufwand, mit dem Benutzer ein bestimmtes Ziel erreichen". Zufriedenheit wird definiert als „Freiheit von Beeinträchtigungen und positive Einstellungen gegenüber der Nutzung des Produkts" [13].

Dabei beschränkt sich der Begriff Usability im Allgemeinen nicht allein auf Softwareprodukte - So können beispielswiese Alltagsgegenstände gleichermaßen nach ihrer Usability bewertet werden wie Computersysteme oder Webseiten [46].

User Experience

Die User Experience betrachtet nicht allein die Gebrauchstauglichkeit eines Systems, sondern schließt die während der Nutzung vom Anwender empfundenen Emotionen mit ein. Dazu zählen positive und negative Gefühle sowie Wahrnehmungen, Meinungen oder auch psychologische und physische Reaktionen. Deshalb wird die User Experience auch als ein Produkt von Werbung, Medien, Verpackungen etc. gesehen, d.h. die empfundenen Emotionen während der Anwendung können durch ebendiese Aspekte beeinflusst werden.

Eine gute Usability allein führt also nicht automatisch zu einer guten User Experience. So erzielen beispielsweise Geräte wie Tablets oder Smartphones trotz teilweise schlechter Usability (beispielsweise spiegelnde Displays) dennoch eine hohe User Experience, da sie neben den praktischen Aspekten auch als Statussymbol und Ausdruck eines Lebensgefühls gelten. In Abbildung 3.1 wird das Zusammenspiel von Usability und User Experience anschaulich dargestellt.

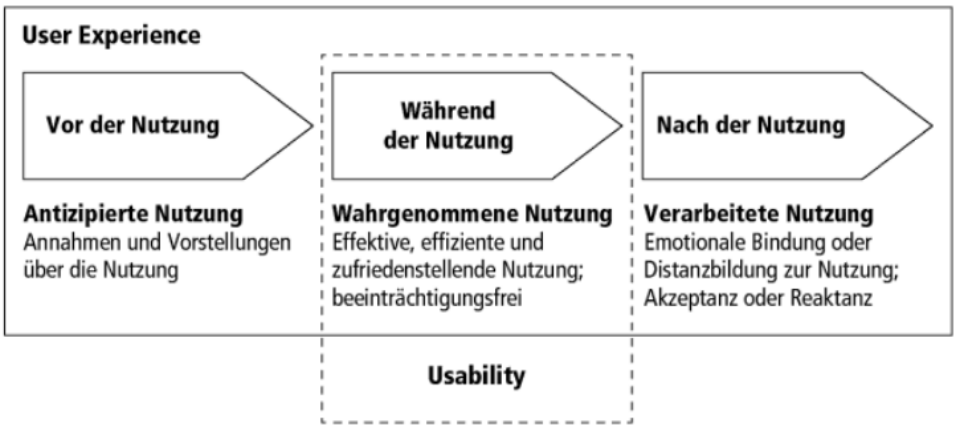


Abbildung 3.1 — Usability und User Experience aus Sicht der ISO 9241 [46].

Bei Betrachtung wird klar, dass Usability also nur einer von mehreren Faktoren ist, der die User Experience eines Produkts beeinflusst [46]. Dabei liegt der Fokus auf den Empfindungen des Anwenders während der Nutzung. Den gesamten Prozess, der die Erwartungshaltungen der Nutzer vor der Anwendung sowie die erlebten Emotionen nach der Benutzung mit einschließt, wird unter dem Begriff User Experience zusammengefasst.

User Interface

Das User Interface stellt die Schnittstelle dar, über die ein Nutzer mit einer Anwendung interagiert. Die Gestaltung des User Interfaces spielt eine tragende Rolle bei der Bewertung der Usability von computergestützten Systemen. Die DIN EN ISO 9241 definiert das User Interface als

> "alle Bestandteile eines interaktiven Systems (Software oder Hardware), die Informationen und Steuerelemente zur Verfügung stellen, die für den Benutzer notwendig sind, um eine bestimmte Arbeitsaufgabe mit dem interaktiven System zu erledigen" [13].

3.2 Kognitionspsychologische Grundlagen

Um die Usability von Systemen bewerten zu können, muss man die Interessen und inneren Vorgänge der Anwender verstehen, um darauf aufbauend Lösungen zu entwickeln. Deshalb spielt bei der Gestaltung solcher Systeme auch immer die menschliche Kognition eine große Rolle. Darüber hinaus sollte auch der größere Kontext der Arbeitsabläufe und -aufgaben betrachtet werden, die mit dem System in Verbindung stehen. Aus diesem Kontext ergeben sich dann die notwendigen Funktionalitäten eines technischen Systems [46].

In der Mensch-Computer-Interaktion existieren laut Norman [34] zwei Hürden, die es zu überwinden gilt: Der *Gulf of execution* (Ausführungskluft), der besagt, dass der Anwender dem System durch die Eingabe entsprechender Befehle seine Absichten vermitteln muss, und der *Gulf of evaluation* (Auswertungskluft), bei dem es darum geht, dass der Anwender den Output eines Systems richtig interpretiert und dementsprechend in seine Handlungen integriert. Grundlage dafür ist immer die Wahrnehmung des Nutzers, also wie er Informationen aufnimmt und verarbeitet. Dabei spielen jedoch auch die umgebenden Einflüsse eine Rolle, die sich auf die Aufmerksamkeit des Anwenders auswirken können. Darüber hinaus sollte es möglich sein, die aufgenommene Information auch im Gedächtnis behalten zu können. Daraus lassen sich drei Hauptaspekte der Kognitionspsychologie, die bei der Interaktion mit einem technischen System relevant sind, ableiten: *Wahrnehmung*, *Aufmerksamkeit* und *Gedächtnis* [38]. Darauf basierend ergeben sich dann Richtlinien, wie Informationen innerhalb eines interaktiven Systems möglichst effektiv dargestellt werden können.

3.2.1 Wahrnehmung

Bei der Gestaltung von User Interfaces spielt primär die visuelle Wahrnehmung eine Rolle. Die Unterstützung durch akustische oder taktile Stimuli erfolgt eher selten. Der menschliche Sehsinn ist durch eine hohe Selektivität geprägt; das bedeutet, dass man im Wesentlichen nur das sieht, worauf man auch achtet. Dieses als *inattentional blindness* (Unaufmerksamkeitsblindheit) bezeichnete Phänomen sollte bei der Gestaltung von Benutzerschnittstellen berücksichtigt werden, da es zeigt, wieviel Einfluss die Darstellung von wichtiger Information auf die visuelle Sinnesverarbeitung haben kann. Zusätzlich kann ein und diesselbe dargestellte Information von verschiedenen Personen unterschiedlich interpretiert werden.
Um solchen Effekten entgegengewirken, wird beim Design von Benutzerschnittstellen verstärkt auf den richtigen Einsatz von passenden Farben und Formen sowie deren Anordnung geachtet, indem die sogenannten *Gestaltgesetze* berücksichtigt werden [38].

3.2.2 Aufmerksamkeit

Die menschliche Aufmerksamkeit ist eine begrenzte kognitive Ressource, das heißt, man kann nie alle Sinneseindrücke, die einen umgeben, gleichzeitig wahrnehmen. Deshalb wird hier zwischen *selektiver* und *geteilter Aufmerksamkeit* unterschieden: Bei ersterer wird die Aufmerksamkeit auf einzelne Reize gerichtet, während bei der geteilten Aufmerksamkeit mehrere Prozesse gleichzeitig berücksichtigt werden können. Aufmerksamkeit ist jedoch kein statischer Zustand: Durch gewisse Reize kann schnell zwischen selektiver und geteilter Aufmerksamkeit gewechselt werden. Die wichtigsten Einflussgrößen sind hier zum einen Bewegung und - bei der akustischen Wahrnehmung - Geräusche, die einen Aufmerksamkeitswechsel hervorrufen. In Bezug auf die Gestaltung von Benutzerschnittstellen sollte die Aufmerksamkeit insofern unterstützt werden, dass

1. der Nutzer während der Durchführung einer Aufgabe fokussiert bleibt, die Ablenkung also möglichst gering gehalten wird und
2. nach einer Unterbrechnung die Aufgabe wieder möglichst problemlos aufgenommen werden kann.

Ein System sollte also so gestaltet sein, dass die Aufmerksamkeit eines Nutzers gezielt auf wichtige Informationen gelenkt wird, die für die Durchführung einer Aufgabe relevant sind. Gleichzeitig sollten nebensächliche Aspekte nur unauffällig bzw. durch aktive Selektion des Nutzers dargestellt werden.

3.2.3 Gedächtnis

Die Art und Weise, wie Informationen und wahrgenommene Reize verarbeitet werden, beruht auf Erfahrung und Wissen, was beides im menschlichen Gedächtnis gespeichert wird. Im Wesentlichen besteht dieses aus drei Teilen: *Sensorischer Speicher*, *Arbeitsgedächtnis* und *Langzeitgedächtnis*. Abbildung 3.2 veranschaulicht deren Zusammenspiel.

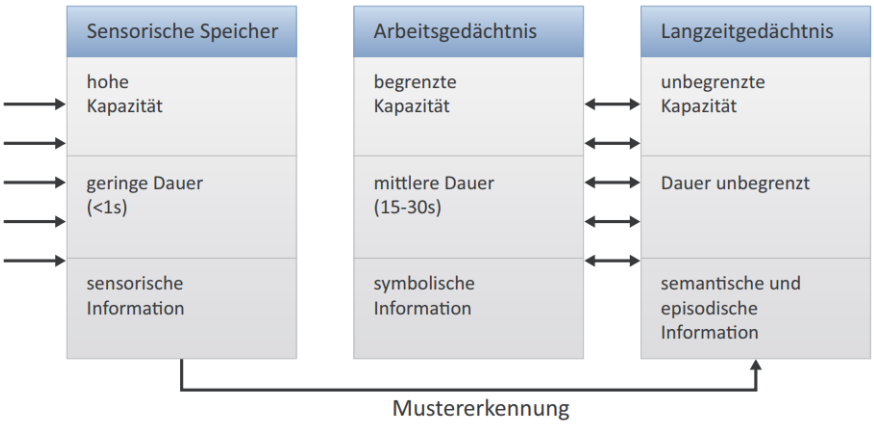


Abbildung 3.2 — Das Zusammenspiel der drei Informationsspeicher im menschlichen Gedächtnis [56].

Der sensorische Speicher nimmt die aus der Umgebung wahrgenommenen Reize kurzzeitig auf und gleicht diese mit den im Langzeitgedächtnis gespeicherten Informationen ab. Um nun hinsichtlich des aufgenommenen Reizes eine Entscheidung zu treffen, greift das Langzeitgedächtnis auf das Arbeitsgedächtnis des Nutzers zu, welches die Ziele und Erwartungen des Nutzers repräsentiert [56].

Das Hauptproblem für die Gestaltung interaktiver Systeme ist die geringe Dauer der Speicherung im sensorischen Gedächtnis. Das bedeutet, dass ein eingehender Reiz nur für einen kurzen Zeitraum zur Verarbeitung zur Vefügung steht. Deshalb sollten Informationen so dargestellt werden, dass sie möglichst unmittelbar verwendet werden können. Darüber hinaus sind lange Antwortzeiten seitens des Systems zu vermeiden.

3.2.4 Human Processor Model

Das Human Processor Model, welches 1983 von Card, Moran und Newell [8] entwickelt wurde, beschreibt die Interaktion zwischen Mensch und Computer, indem die drei Aspekte Wahrnehmung, Aufmerksamkeit und Gedächtnis als eigene Prozessoren bei der menschlichen Informationsverarbeitung verstanden werden. Dabei sind drei Subsysteme von Bedeutung:

1. Perzeptuelles System;
2. Kognitives System;
3. Motorisches System.

Das Zusammenspiel dieser Systeme kann auf zwei Arten erfolgen: Zum einen seriell, wenn auf einen eigehenden Reiz eine bestimmte Reaktion erforderlich ist. Für die Informationsverarbeitung bedeutet das, dass ein eingehender Stimulus zunächst vom perzeptuellen System aufgenommen wird. Dabei kann es sich beispielsweise um einen visuellen Reiz handeln. Die Aufnahme in das perzeptuelle System erfolgt dann über das menschliche Auge, welches die Information an das kognitive System weiterleitet. Hier befindet sich das Gedächtnis, auf dessen darin enthaltene, gespeicherte Informationen zugegriffen wird. Das kognitive System gleicht die aufgenommene Information mit dem darin enthaltenen Kurz- und Langzeitgedächtnis ab und entscheidet dann, welcher Reiz an das motorische System weitergegeben werden soll, das eine Reaktion beim Menschen auslöst. Je nach eingehendem Reiz und Verarbeitung können dann beispielsweise bestimmte Bewegungen, aber auch Sprache ausgelöst werden. Bei bestimmten Tätigkeiten können jedoch auch alle drei Systeme parallell operieren. Beim Autofahren beispielsweise kann der Fahrer gleichzeitig die Steuerung übernehmen sowie Musik hören oder sich mit dem Beifahrer unterhalten [1]. Abbildung 3.3 stellt die Interaktion der drei Subsysteme anschaulich dar.

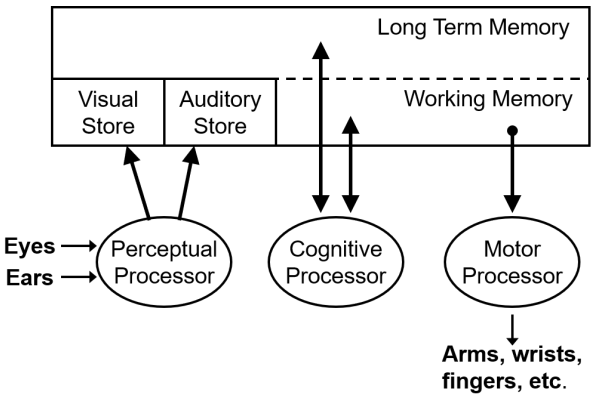


Abbildung 3.3 — Das Human Processor Model zur Beschreibung der menschlichen Informationsverarbeitung [1].

3.3 Implikationen für die Mensch-Computer-Interaktion

3.3.1 Gestaltgesetze

Auf Basis der Kenntnisse über die menschliche Informationsverarbeitung, wie sie in den letzten Abschnitten beschrieben wurden, lassen sich diverse Guidelines ableiten, die ein technisches System erfüllen sollte, um eine hohe Usability zu erzielen. Die Gestaltgesetze liefern eine gute Orientierung darüber, wie einzelne Bedienelemente platziert werden sollten, um den Nutzer bestmöglich zu unterstützen (Abbildung 3.4).

1. **Gesetz der Nähe:** Objekte, die nahe beieinander angeordnet sind, werden als zusammengehörig wahrgenommen. Eine solche Anordnung kann bewusst für Gruppierungen genutzt werden, was den Einsatz von anderen Elementen wie Trennlinien reduziert.
2. **Gesetz der Ähnlichkeit:** Ähnliche Objekte werden ebenfalls vom Auge gruppiert; die Ähnlichkeit kann dabei durch Farbe, Form, Größe, Textur oder Bewegungsrichtung sichtbar werden.
3. **Gesetz der Prägnanz:** Bei mehreren Objekten werden diejenigen zuerst wahrgenommen, die sich in gewisser Weise vom Rest abheben. Dieses Gestz eignet sich zur Hervorhebung wichtiger Informationen.
4. **Gesetz der Symmetrie:** Symmetrisch zueinander angeordnete Objekte werden stärker wahrgenommen als zufällig angeordnete Elemente. Dieses Gesetz eignet sich zur bewussten Lenkung von Aufmerksamkeit.
5. **Gesetz der Verbundenheit:** Elemente, die beispielsweise durch Verbindungslinien miteinander verbunden sind, werden als zusammengehörig wahrgenommen und eignen sich sehr gut als Mittel zur Gruppierung.
6. **Gesetz der Geschlossenheit:** Die Form eines Objekts muss nicht zwingend komplett dargestellt werden, da das menschliche Auge fehlende Teile einer Figur automatisch komplettiert. Dies kann beispielsweise zur Reduktion der visuellen Komplexität genutzt werden.

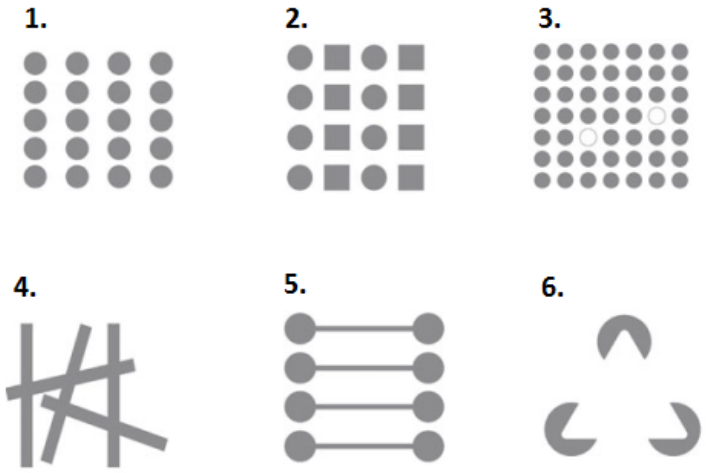


Abbildung 3.4 — Beispiele für die sechs Gestaltgesetze [30].

Bei der Gestaltung technischer Systeme sollte der Einsatz der Gestaltgesetze berücksichtigt werden, um hier Usability-Problemen vorzubeugen. Bei der Evaluation können sie gegebenenfalls als Grundlage für Verbesserungen genutzt werden.

3.3.2 Metaphern

Neben dem Anwenden der Gestaltgesetze können Metaphern dabei helfen, visuelle Informationen auf effektive Weise darzustellen. Metaphern dienen dazu, im Kopf des Anwenders gewisse, vertraute Bilder zu erzeugen und helfen somit, Analogien zur Wirklichkeit herzustellen, um Strukturen und Funktionen schneller zu verstehen. Metaphern sind aus dem informationstechnlogischen Umfeld kaum noch wegzudenken. Moderne Betriebssysteme nutzen Metaphern wie *Ordner* oder *Papierkorb* und erleichtern dem Nutzer somit die Orientierung und Interaktion [11]. Bei der Wahl geeigneter Metaphern sollte jedoch auch immer die anvisierte Zielgruppe berücksichtigt werden, da unpassende oder stark abstrakte Metaphern auch Verwirrung beim Anwender auslösen können [52].

Durch das Aufkommen von 3D und VR wurden immer weitere Metaphernansätze entwickelt. So dienen ganze Städte oder Sonnensysteme dazu, die Komplexität von interaktiven Systemen zu reduzieren und sie dem Nutzer dadurch leichter zugänglich zu machen. Gerade im Bereich der Informations- bzw. Softwarevisualisierung ist dies extrem hilfreich, was in Kapitel 4 noch näher betrachtet wird.

3.4 Methoden der Usability Evaluation

Zur Evalulation von Usability wird für gewöhnlich zwischen verschiedenen Standards gewählt. Dabei unterscheidet man grob zwischen zwei Gruppen: empirische und analytische Methoden. Bei den empirischen Methoden werden Aussagen über die Usability eines Systems durch Befragung und Beobachtung der tatsächlichen Nutzer getroffen. Analytische Methoden werden von einem Usability-Experten alleine durchgeführt, indem er versucht, sich in den Nutzer hineinzuversetzen und so Entscheidungen über die weitere Entwicklung der Anwendung zu treffen. Im Folgenden sollen beide Evaluationsmethoden im Ansatz beschrieben werden.

Analytische Methoden

Bei den analytischen Evaluationsmethoden werden keine Versuchspersonen involviert, das heißt, das zu evaluierende System wird ausschließlich vom zuständigen Usability-Experten getestet. Dabei stehen ihm folgende Methoden zur Verfügung:

- Heuristische Evaluation
- Design-Guidelines
- Formal-analytische Verfahren

Bei der Heuristischen Evaluation werden eine Reihe von Usability-Prinzipien, sogenannte Heuristiken, auf deren Einhaltung geprüft. Diese Heuristiken weisen auf bestimmte Problemkategorien bei der Gestaltung von interaktiven Systemen hin [33] und beschreiben erwünschte Eigenschaften der Interaktion zwischen dem Anwender und dem System [22]. Dieses Verfahren wird von einem oder mehreren Usability-Experten durchgeführt, die versuchen, sich in die Rolle des Nutzers hineinzuversetzen. Dabei steht das Ziel im Vordergrund, die Sichtweise eines Nutzers aus der Zielgruppe einzunehmen.

Design-Guidelines stellen Richtlinien zur Gestaltung von Schnittstellen interaktiver Systeme dar, das heißt, sie stehen am Anfang des Gestaltungsprozesses. Diese Richtlinien bestehen aus einer Auflistung von Grundsätzen, die bei der Entwicklung und Gestaltung von interaktiven Systemen berücksichtigt werden sollten. Das vereinfacht dem Entwickler Designentscheidungen und ermöglicht dem Kunden bzw. dem Usability-Experten eine objektive Bewertung der Applikation. Dabei können Guidelines sowohl formativ als auch summativ angewandt werden, was bedeutet, dass sie sich sowohl während des Entwicklungsprozesses als auch am Schluss zur Gesamtbewertung des Systems eignen. Solche Design-Guidelines sind beispielsweise die bereits beschriebenen Gestaltgesetze.

Bei den formal-analytischen Verfahren wird zwischen zwei Hauptgruppen unterschieden, nämlich den aufgabenanalytischen Verfahren und den Expertenleitfäden. Aufgabenanalytische Verfahren basieren auf mit dem System zu erfüllende Aufgaben, die von den Usability-Experten von Teilaufgaben bis zu konkreten Handlungen aufgeschlüsselt und mit einer formalen Modellsprache beschrieben werden. Dadurch ist es möglich, die einzelnen Schritte sowohl einzeln als auch kumuliert anhand verschiedener Metriken, wie beispielsweise Ausführungzeiten, zu bewerten. Ein großer Vorteil der formal-analytischen Methoden liegt darin, dass Systeme bereits in Form der

Lastenhefte zu Beginn des Entwicklungsprozesses evaluiert werden können. Die Implementierung von Nutzerschnittstellen oder Funktionen ist deshalb für die Evaluation nicht nötig. Ein bekannter Ansatz dieser Methode ist das sogenannte GOMS-Modell [8], bei dem die Nutzerinteraktion mit dem System in einzelne Teilaufgaben unterteilt wird.

Expertenleitfäden wiederum sind so konzipiert, dass das System nicht aufgrund seiner Eignung zur Aufgabenerfüllung untersucht wird, sondern aus der Perspektive der Software-Ergonomie. Auch Expertenleitfäden werden meist früh im Software-Entwicklungsprozess angewendet. Dadurch sollen quantitative Aussagen über gewisse Aspekte der Gestaltung von Benutzerschnittstellen getroffen werden können, die sich jedoch nicht auf allgemeine Usability-Probleme, sondern vielmehr auf einzelne (fehlende) Funktionen innerhalb des Systems beziehen [46].

Empirische Methoden

Bei empirischen Methoden wird die spätere intendierte Nutzergruppe mit in den Entwicklungsprozess einbezogen. Das bedeutet, dass durch die Interaktion des Anwenders mit dem System bereits dessen Schwachstellen aufgedeckt werden sollen, um darauf aufbauend Verbesserungsvorschläge zu entwickeln. Die zu dieser Gruppe gehörenden Verfahren lauten

- Walkthrough-Verfahren
- Nutzungstagebücher
- Usability-Tests und
- Fragebögen.

Im Folgenden sollen die oben genannten Verfahren jeweils grob skizziert werden.

Bei Walkthrough-Verfahren wird einer Nutzergruppe der Vorschlag eines Systems vorgelegt, den sie anhand zuvor festgelegter Kriterien bewerten muss. Mit Vorschlag ist gemeint, dass es sich um keinen funktionierenden Prototypen handelt, sondern vielmehr um Beschreibungen der geplanten Funktionen und Bedienelemente. Oft werden hierfür auch Mock-Ups eingesetzt. Ein bekannter Vertreter dieses Verfahrens ist der Cognitive Walkthrough, bei dem überprüft wird, wie einfach Nutzer eine Aufgabe mit dem System lösen können, ohne zuvor umfassende Anweisungen oder eine Einführung erhalten zu haben [37].

Nutzungstagebücher gehören prinzipiell der Gruppe der Fragebögen an, unterscheiden sich aber in ihrem Betrachtungszeitraum und -kontext. Sie werden verwendet, um ein System über einen längeren Zeitraum zu bewerten. Darüber hinaus wird das System schon unter Realbedingungen getestet, also im alltäglichen Gebrauch und nicht in einem Testlabor. Bei Usability-Tests wird dem Anwender eine Reihe von Tasks vorgelegt, die er unter Beobachtung eines Usability-Experten durchführen soll. Dabei werden verschiedene Metriken gemessen und vom Beobachter festgehalten. Fragebögen dienen der Erfassung subjektiver Einstellungen des Nutzers dem System gegenüber [46]. Dabei existiert eine Vielzahl an standardisierten Fragebögen, die speziell auf die Usability eines Systems abzielen. Beide Methoden sind Bestandteil der in dieser Arbeit durchgeführten Studie und werden in Kapitel 5 noch umfassender beschrieben.

Neben den bereits aufgeführten Methoden gibt es noch diverse Untergruppen, die hier nicht aufgelistet sind. Generell lässt sich im Bereich der empirischen Methoden zwischen Behavioral und Attitudinal Testing unterscheiden: Attitudinal Testing erfasst qualitative Aussagen über die Gefühle und Gedanken des Nutzers. Beim Behavioral Testing hingegen werden quantitative Daten gesammelt und der Fokus liegt auf der Interaktion mit dem System. Abbildung 3.5 veranschaulicht diese Aufteilung: Zusätzlich wird hier noch beim Nutzungskontext des Systems differenziert. Wie bereits erwähnt, werden viele Methoden in eigens dafür eingerichteten Usability-Laboren durchgeführt. Ein Beispiel dafür sind Eyetracking-Studien, bei denen die Blickbewegungen des Nutzers aufgezeichnet werden. Auch Usability-Studien werden für gewöhnlich in eigenen Laboren durchgeführt. Bestimmte Systeme hingegen können unter realen Bedingungen getestet werden, beispielsweise direkt vom Anwender zuhause. Sinnvoll ist diese Form der Usability-Evaluierung beispielsweise für Systeme im Bereich Ubiquitous Computing, die Anwender in ihrer alltäglichen Arbeit unterstützen sollen.

Darüber hinaus finden auch Methoden Anwendung, in welchen das eigentliche System gar nicht involviert wird. Ein Beispiel hierfür wäre das Card Sorting: Hier werden für die Entwicklung der Menüstruktur Papierkarten verwendet, die von den Probanden in eine sinnvolle Reihenfolge gebracht werden sollen. Solche Methoden eignen sich jedoch nur in sehr frühen Phasen der Systementwicklung [43].

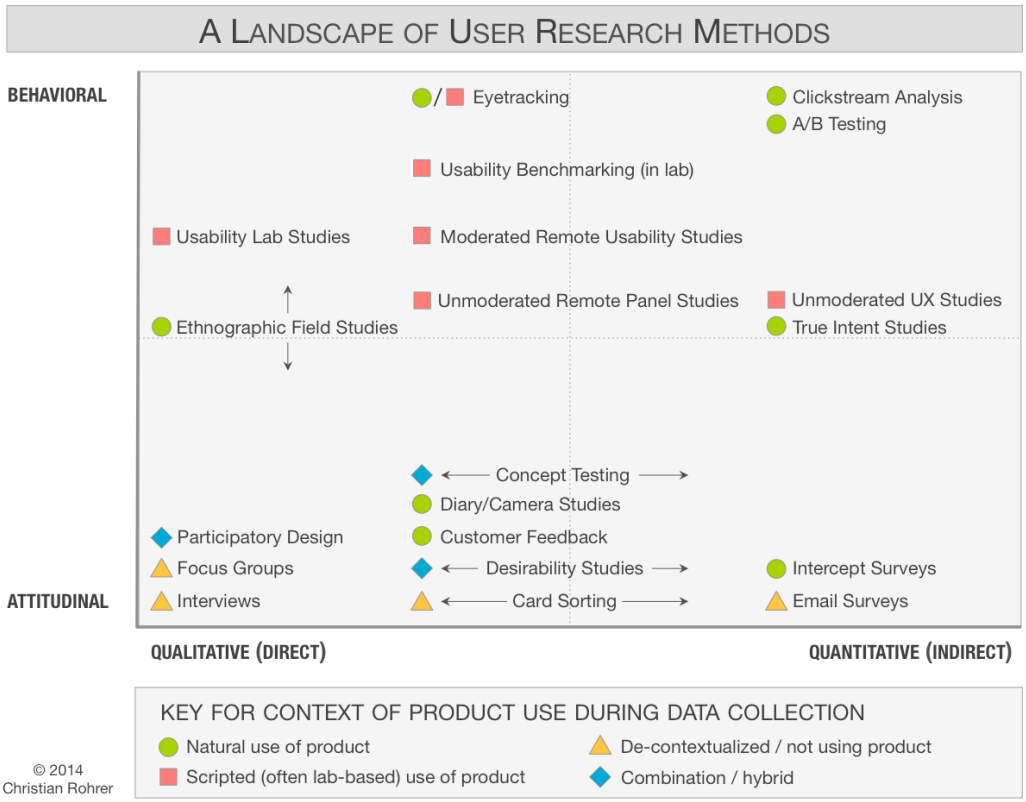


Abbildung 3.5 — Übersicht über verschiedene Methoden der Usability-Evaluation [43].

4 Softwarevisualisierung

4.1 Grundlagen

Trotz des Prinzips der Modularisierung, welches bereits in Kapitel 2 vorgestellt wurde, sind große Softwareprojekte mit einer hohen Zahl an Lines of Code meist sehr schwer zu erfassen, da dadurch der Umfang und die Komplexität eines Programms schnell ansteigen kann. Außerdem fehlt es gerade für Laien an Möglichkeiten, Software verstehen zu können. Da aber im heutigen Berufsalltag viele Teams interdisziplinär arbeiten und die Softwareentwicklung in allen Bereichen immer präsenter wird, müssen Möglichkeiten gefunden werden, die Software verständlich und realitätsnah darstellen können. Bekannt ist, dass bildliche Darstellungen vom menschlichen Gehirn einfacher verarbeitet werden können als reiner Text - dadurch kann der Prozess der Verständnisgewinnung unterstützt werden [57]. Das Gebiet der Softwarevisualisierung vereint Aspekte aus den Bereichen Informationsvisualisierung, Software Engineering, Mensch-Computer-Interaktion, Grafikdesign und der Kognitionspsychologie [26].

Softwarevisualisierungen helfen dabei, Informationen auf effektive Art und Weise zu kommunizieren und ermöglichen dem Nutzer Einblick in Systeme und Prozesse [53]. Dies ist vor allem auch deshalb wichtig, da mit der steigenden Komplexität eines Softwaresystems gleichzeitig der Aufwand wächst, das System zu warten und weiterzuentwickeln [5]. Um diese Aufwandskosten also möglichst gering zu halten, ist es von hohem Interesse, den Prozess der Verständnisgewinnung über ein Softwaresystem zu vereinfachen und zu beschleunigen. Nach Maletic et al. [25] ist es allerdings unmöglich, mit nur einem Visualisierungstool alle gewünschten Bereiche der Software abzubilden. Um also eine passende Visualisierungsart zu finden, ist es zunächst nötig, die wichtigsten Aspekte der Visualisierung zu identifizieren. Dabei sind im Wesentlichen fünf Dimensionen von Bedeutung:

Ziel: Durch die Festlegung eines Ziels werden die Daten, die mit der Softwarevisualisierung übermittelt werden sollen, definiert. Das können beispielsweise ganze Systeme, aber auch nur Teile davon sein. Auch die reine Betrachtung von Abhängigkeiten oder Beziehungen ist möglich.

Nutzer: Bei der Wahl der zu visualisierenden Information und der Darstellungsform muss auch immer der Nutzer berücksichtigt werden. Dies können beispielsweise Entwickler, Projektleiter, Manager oder Kunden sein.

Aufgabe: Die Aufgabe ist eng verknüpft mit dem Ziel und definiert den Zweck der Visualisierung. Zu möglichen Aufgaben zählen Reverse-Engineering, Entwicklung, Wartung oder Controlling.

Repräsentation: Die Dimension der Repräsentation legt fest, mithilfe welcher grafischen Elemente die Informationen dargestellt werden sollen.

Medium: Hier wird definiert, durch welches Medium die Visualisierung erfolgen soll. Das kann beispielsweise 2D oder 3D sein. Auch der jeweilige Träger spielt eine Rolle, also ob Papier, ein Bildschirm oder eine VR-Brille verwendet wird.

4.1.1 Anwendungsbereiche

Im Allgemeinen lässt sich zwischen drei Aufgabenbereichen in der Softwarevisualisierung unterscheiden, in denen die Verständnisgewinnung über ein Softwaresystem eine tragende Rolle spielt:

1. Konzeptionierung und Entwicklung von neuartigen Softwaresystemen;
2. Wartung und Optimierung bestehender Softwaresysteme;
3. Dokumentation des Software-Entwicklungsprozesses.

Für die Konzeptionierung und Entwicklung neuer Software finden häufig Visualisierungen Einsatz, die auf Systemodellierungen beruhen. Dabei werden in Diagrammform statische Aspekte der Softwarearchitektur, dynamische Aspekte der Systemausführung und physische Aspekte der Systemoperationalisierung dargestellt. Die Erstellung solcher Modelle geschieht bereits durch den Entwickler, der sie dann mit der Implementierung umsetzt. Ein bekanntes Beispiel für diesen als *Forward Engineering* bezeichneten Bereich der Softwarevisualisierung ist die *Unified Modeling Language* (UML).

Die Wartung und Optimierung von bereits bestehenden Softwaresystemen wird als *Reverse Engineering* bezeichnet. Dabei wird oft auf sogenannte Refactoring-Methoden zurückgegriffen: Die Struktur des Quellcodes kann somit verbessert werden, ohne dass der Programmablauf dadurch beeinflusst wird. Dieses Verfahren wird angewandt, um schlechte Strukturen in der Architektur zu entdecken und auszubessern. Passende Visualisierungstechniken können hier helfen, automatisch Informationen über Strukturen, Abhängigkeiten und dynamische Aspekte zu generieren und somit zur Verständnisgewinnung über ein Softwaresystem beizutragen.

Zuletzt können noch geeignete Visualisierungstechniken dazu beitragen, Aufschluss über die zeitliche Entwicklung eines Softwaresystems zu gewinnen. Das geschieht üblicherweise durch die Verknüpfung von Daten, die aus dem Quellcode gewonnen werden können, mit Informationen über die zeitliche Entwicklung des Softwaresystems. Dies ist vor allem dann hilfreich, um Hilfestellungen für zukünftige Entscheidungen in der Softwareentwicklung zu liefern [5].

4.1.2 Herausforderungen

Die größte Herausforderung bei der Entwicklung von Softwarevisualisierungen liegt darin, geeignete Abbildungen mithilfe von visuellen Metaphern zu finden [18]. Denn die gewählte Darstellung entscheidet maßgeblich mit darüber, ob es beim Anwender zu einer Reduzierung der Komplexität kommt: So fanden Knight und Munro [21] heraus, dass zwar beispielsweise einfache Graphdarstellungen Einblicke in den Aufbau eines Systems gewähren, es allerdings nicht schaffen, die Komplexität aufzulösen. Darüber hinaus ist die häufig sehr hohe Datenmenge eine weitere Herausforderung, die bei der Repräsentation berücksichtigt werden muss. Deshalb ist

es nicht zielführend, ein komplexes Softwaresystem in nur einer Darstellung zu repräsentieren, vielmehr sollte es auf mehrere verschiedene Darstellungen, sogenannte *Views*, aufgeteilt werden, um die Informationen gebündelt zu vermitteln [28]. Ein weiterer Vorteil liegt darin, dass sich der Anwender durch die so entstehende Interaktion bzw. Exploration das Dargestellte besser einprägt [27].

4.1.3 Qualitätskriterien

Young und Munro [61] entwickelten einen Anforderungskatalog an Eigenschaften, die bei der Entwicklung von Softwarevisualisierungen berücksichtigt werden sollten, um ein positives Nutzungserlebnis zu gewährleisten.

1. **Einfache Navigation:** Die Visualisierung sollte Funktionen anbieten, die den Nutzer dabei unterstützen, durch die Anwendung zu navigieren. Das Ziel dabei ist, es dem Nutzer zu ermöglichen, sich schnell und einfach innerhalb der Anwendung zurecht zu finden. Darüber hinaus soll Desorientierung vermieden werden.
2. **Hoher Informationsgehalt:** Softwarevisualisierungen sollten soviele Informationen wie möglich darstellen, ohne dabei den Nutzer zu überfordern.
3. **Geringe visuelle Komplexität:** Die strukturelle Komplexität ist abhängig von der Komplexität der dargestellten Information. Durch die Aufteilung der Visualisierung in mehrere Views soll dieses Problem verringert werden.
4. **Verschiedene Levels of Detail:** Der Nutzer soll die Möglichkeit haben, die Visualisierung sowohl im Ganzen als auch auf feingranularer Ebene erforschen zu können.
5. **Stabilität gegenüber Veränderungen:** Kleinere Veränderungen oder Anpassungen der dargestellten Information sollten nicht in größeren Veränderungen der Visualisierung resultieren, da dies zu einer Desorientierung beim Nutzer führen kann.
6. **Verwendung passender Metaphern:** Metaphern helfen dabei, sich schnell in der Visualisierung zurecht zu finden. Außerdem eignen sie sich als ein guter Startpunkt zur ersten Verständnisgewinnung über ein Softwaresystem.
7. **Zugängliches User Interface:** Das User Interface sollte flexibel genug sein, um eine intuitive Bedienung zu ermöglichen. Dieser Aspekt bezieht sich auch auf die zur Exploration der Visualisierung benötigte Hardware, die nicht zu sehr vom Standard abweichen sollte.
8. **Einbindung weiterer Informationsquellen:** Durch Softwarevisualisierungen können bestimmte Informationen auf andere Art dargestellt werden. Dennoch sollte es möglich sein, innerhalb der Anwendung zwischen der Visualisierung und der Ursprungsinformation, wie sie beispielsweise im Source Code vorliegt, zu wechseln.
9. **Möglichkeit zur Interaktion:** Dem Nutzer sollte es möglich sein, auf mehrere Arten mit der Anwendung zu interagieren. Das vereinfacht die Verständnisgewinnung und hilft dabei, die aufgenommenen Informationen zu verinnerlichen.
10. **Automatisierung:** Zusätzlich kann durch Automatisierung bestimmter Funktionen oder Prozesse eine höhere Praktikabilität erzielt werden.

4.2 Arten von Softwarevisualisierungen

Zur graphischen Darstellung von Softwaresystemen existieren verschiedene Ansätze, die unterschiedliche Aspekte der Software abdecken. Caserta und Zendra liefern mit Abbildung 4.1 eine umfangreiche Übersicht über den aktuellen Forschungsstand im Bereich der Softwarevisualisierungen. Dabei unterteilen sie die verschiedenen Techniken hinsichtlich ihrer Granularität in drei Abstraktionsbenen:

1. Source Code-Ebene;
2. Package-, Klassen- und Methodenebene;
3. Architekturebene.

	Level	Focus	Visualization Technique	Representation	Year
Time T Visualization	Line	Line properties	Seesoft	2D colored pixel	1992
			Sv3d	3D colored cuboid	2003
	Class	Functioning, Metrics	Class BluePrint	2D layers and graph	1999
	Architecture	Organization	Treemap	2D/3D colored nested boxes	1991
			Circular Treemap	2D/3D colored nested circles	1991
			City/Cities	3D city metaphor	1993
			Sunburst	2D colored radial display	1998
			Solar System	3D solar system metaphor	2003
			Voronoi Treemap	2D colored irregular shapes	2005
		Relationships	Dependency Structure Matrix	2D table	1981
			UML	2D diagrams	1996
			Geon	3D geon diagrams	1998
			Solar System	3D solar system metaphor	2003
			Landscape	3D landscape metaphor	2004
			Hierarchical Edge Bundles	2D graph with bundled edges	2006
			City/Cities	3D city metaphor with edges	2007
			3D Clustered Graph	3D clustered graph	2007
		Metrics	Polymetric views	2D graph	1999
			Solar System	3D Solar system metaphor with edges	2003
			UML MetricView	2D UML diagrams with charts on top	2005
			Treemap metrics	2D nested boxes with color and texture	2005
			City	3D City metaphor	2005
			UML Area Of Interest	2D diagrams with area of interest	2006
Visualizing Evolution	Line	Changes	Code Flow	cable-and-plug wiring metaphor	2007
	Class		TimeLine	3D building metaphor	2008
	Archi.	Organizational Changes	Hierarchical Edge Bundles	2D graph with bundled edges	2008
		Metrics Evolution	Evolution Matrix	2D matrix	2001
			RelVis	2D Kiviat diagrams and graph	2005
			City/Cities	3D city metaphor with animation	2008

Abbildung 4.1 — Übersicht über verschiedene Visualisierungsarten und deren Abstraktionsebenen mit Beispielen [9].

Zusätzlich kategorisierten sie ihre Unterteilung in zwei Gruppen, wie sie bereits in Kapitel 4.1.1 vorgestellt wurden: Zum einen kann ein Softwareprojekt zu einem bestimmten Zeitpunkt T betrachtet werden, zum anderen kann eine Visualisierung aber auch die Evolution der Software darstellen.

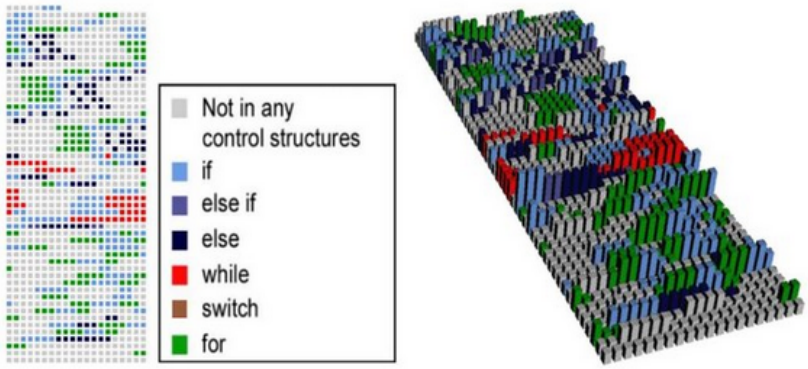
Dabei liegt der Fokus schwerpunktmäßig auf 2D- und 3D-Repräsentationen, die verschiedene Darstellungsansätze verfolgen. Gerade bei sehr umfangreichen Softwareprojekten mit einer großen Menge an darzustellender Information eignen sich 3D-Ansätze, da durch die zusätzliche dritte Dimension eine höhere Zahl an Daten dargestellt werden kann [58]. Besonders kompakte Darstellungen liefern Pixelvisualisierungen, die Software auf Source Code-Ebene abbilden, beispielsweise die 2D-Visualisierung *SeeSoft* [15]. Marcus et al. [27] erweiterten diesen Ansatz um eine dritte Dimension mit *sv3D*. Bei dieser Visualisierungstechnik werden Informationen über Zeilenlänge und Kontrollstrukturen durch eingefärbte Pixel dargestellt (Abbildung 4.2(a)).

Für Strukturen auf höherer Abstraktionsebene, also bei der Visualisierung von Packages, Klassen und Methoden, soll durch geeignete Techniken Verständnis über die innere Funktionsweise solcher Komponenten vermittelt werden. Einige Forschungsarbeiten beschäftigten sich in diesem Bereich beispielsweise mit der Visualisierung von Kohäsion innerhalb einer Klasse [10]. Ducasse und Lanza [14] liefern mit *Class Blueprint* eine einfache Visualisierungstechnik, die die Gesamtstruktur einer Klasse, den Kontrollfluss zwischen den Methoden dieser Klasse sowie Methodenaufrufe darstellt.

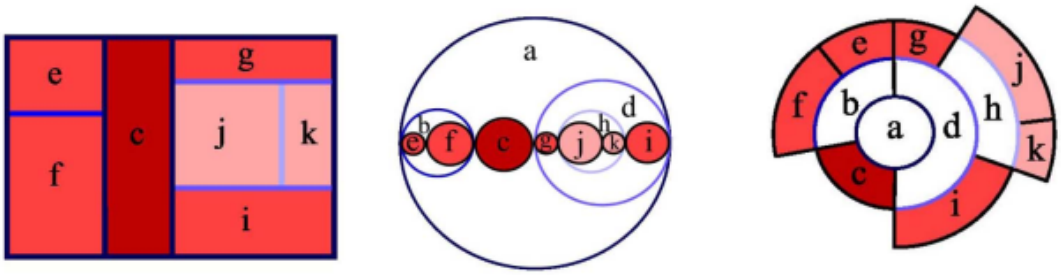
Darstellungen auf Architekturebene stellen den wichtigsten Bereich der Softwarevisualisierungen dar. Hier eignen sich zur Abbildung von Abhängigkeiten und Beziehungen Graphen und Netze (Abbildung 4.2(d)). Da diese Darstellungsmethoden bei umfangreichen Architekturen mit einer hohen Zahl an Abhängigkeiten schnell an ihre Grenzen stoßen, eignen sich verschiedene Arten der sogenannten *Treemap* (Abbildung 4.2(b)). Die Treemap ist im Gegensatz zu graphbasierten Darstellungen platzsparender und wird durch eine Baumstruktur abgebildet, bei welcher die Größe der Knoten besonders prägnant visualisiert werden. Deshalb ist sie auch besonders gut zur Darstellung von Hierarchiestrukturen geeignet.
Bei Visualisierungen mit einer hohen Informationsdichte eignen sich geclusterte Graphen, die eine Reduzierung der Komplexität bewirken. Dabei kommen auch häufig dreidimensionale Graphen zum Einsatz (Abbildung 4.2(c)).

Gerade im dreidimensionalen Bereich werden darüber hinaus gerne *Realwelt-Metaphern* verwendet (Abbildung 4.3). Hier wird anstelle von abstrakten grafischen Elementen die Software mithilfe von Objekten aus der Wirklichkeit dargestellt. Der Einsatz von Metaphern soll eine bessere Zugänglichkeit für den Benutzer sowie eine intuite Interpretation der Informationen bewirken [45]. Das können Landschaften sein, die durch ihre Oberflächenstruktur, beispielsweise Berge oder Täler, und Anordnung Daten aus dem Source Code abbilden (Abbildung 4.3(a)). Yang, Graham and Berrigan [60] entwarfen einen Prototyp, der ein Sonnensystem zur Vorlage hat. Eine solche Darstellung eignet sich für die Repräsentation vieler Abhängigkeiten.

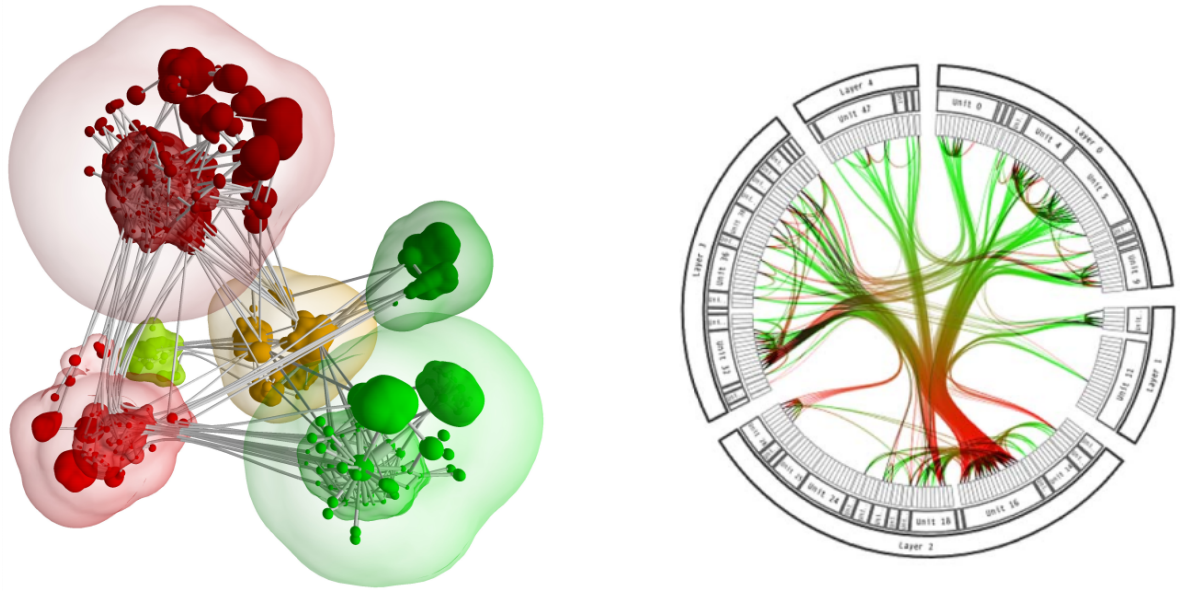
Ein weiterer beliebter Ansatz ist die Stadtmetapher, bei der die einzelnen Softwarekomponenten als Gebäude dargestellt werden. Panas et al. [36] liefern hierfür ein sehr realistisches Beispiel (Abbildung 4.3(c)). Dabei stellen die Gebäude Klassen dar; die Größe der Gebäude gibt die Lines of Code der Klasse wieder. Eine hohe Gebäudedichte an diversen Stellen innerhalb der Stadt impliziert eine hohe Kopplung zwischen den Komponenten. Darüber hinaus liefern zusätzliche Metriken Informationen zu den Klassen: So bestehen die Gebäude aus unterschiedlichen Strukturen, die Aussagen über die Qualität der Implementierung erlauben. Beispielsweise stellen alte oder eingestürzte Gebäude Source Code dar, der überarbeitet werden muss.



(a)

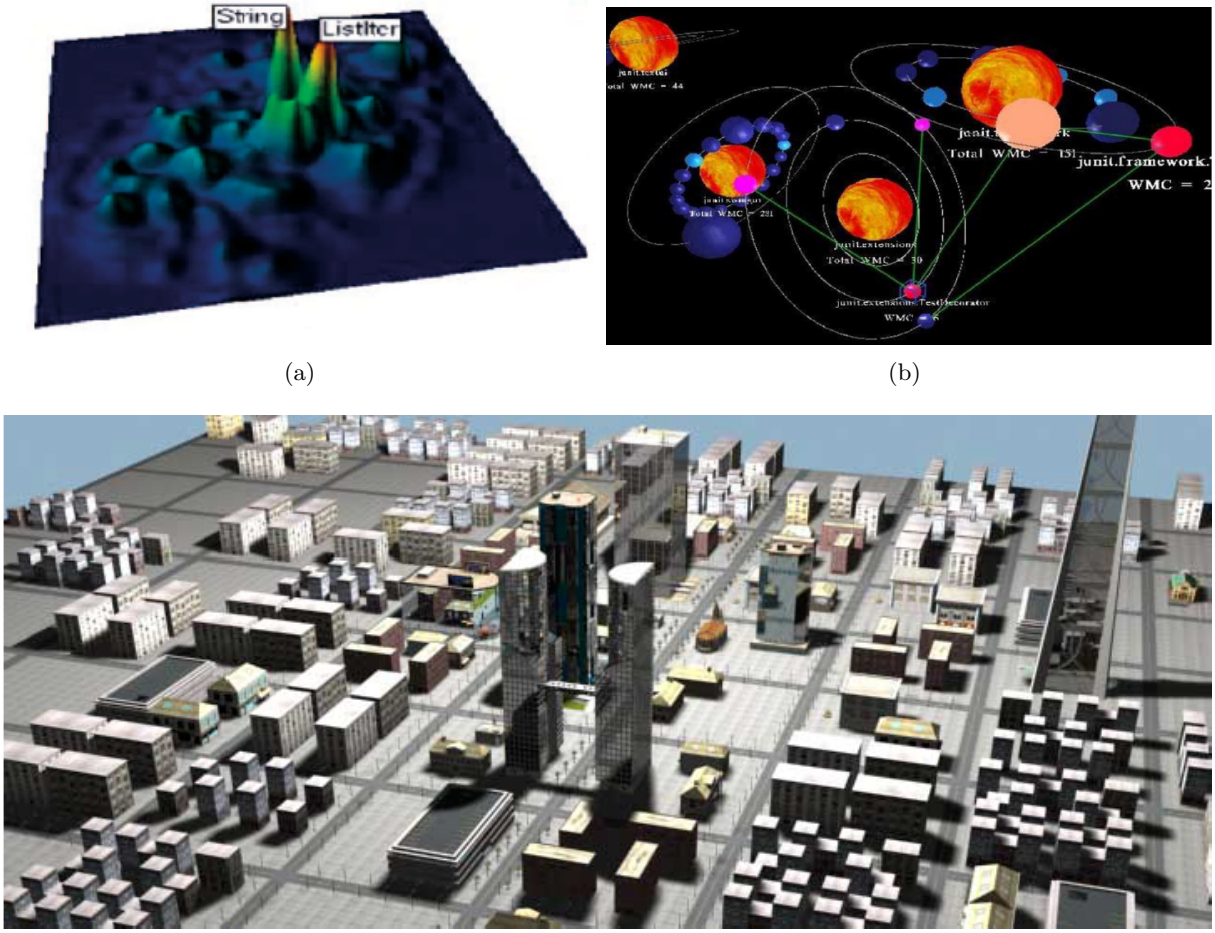


(b)



(c) (d)

Abbildung 4.2 — Beispiele für verschiedene Arten von Softwarevisualisierungen: (a) Softwarevisualisierung auf Source Code-Ebene in 2D und 3D; (b) Unterschiedliche Darstellungen von Treemaps; (c) Geclusterter dreidimensionaler Graph; (d) Netzdarstellung [15, 27, 9, 2, 20].



(a) (b)

(c)

Abbildung 4.3 — Realwelt-Metaphern als Vorlage zur Darstellung von Softwarearchitekturen: Landschaften, Sonnensysteme und Städte [54, 60, 36].

Neben den statischen Bereichen der Software werden aber auch dynamische Aspekte berücksichtigt: Fahrzeuge, die sich in den Straßen bewegen, stellen den Programmfluss dar. Dabei dienen Farbspuren der Orientierung, um die Richtung nachzuverfolgen.

Während also Ansätze für Softwarevisualisierungen in 2D und 3D zahlreich existieren, ist der Bereich VR diesbezüglich bisher weniger gut erforscht. Obwohl VR neben der Unterhaltungsbranche mittlerweile auch in vielen wissenschaftlichen Gebieten Nutzen findet (z.B. Medizintechnik, Flug- und Fahrsimulation etc.), gibt es in der Informations- und Softwarevisualisierung kaum Beispiele, obwohl davon ausgegangen wird, dass die Verständnisgewinnung aufgrund der durch die in der virtuellen Welt hervorgerufene Immersion gesteigert werden kann [24]. Ein weiteres Problem liegt darin, dass die meisten Entwürfe nicht über einen konzeptionellen Status hinausgehen, da es an nachvollziehbaren und prüfbaren Ergebnissen in Form von Prototypen oder Nutzerstudien mangelt [19].

In der vorliegenden Arbeit wird diese Problematik aufgegriffen, indem die Usability zweier Softwarevisualisierungen evaluiert wird. Es wird ein auf VR basierender Ansatz, repräsentiert durch eine Inselmetapher, untersucht. Als Vergleich dient eine Visualisierung in 2D, die durch einen kräftebasierten Graphen dargestellt wird. Im folgenden Abschnitt sollen beide Visualisierungsarten vorgestellt werden.

4.3 Forschungsgegenstand

Bei den zu evaluierenden Visualisierungen handelt es sich um zwei vom Deutschen Zentrum für Luft- und Raumfahrt, Einrichtung für Simulations- und Softwaretechnik, entwickelten Ansätze in 2D und VR. Da sich die Forschung bisher überwiegend mit Vergleichen zwischen 2D und 3D bzw. 3D und VR beschäftigt hat, stellt diese Arbeit einen innovativen Forschungsrahmen dar. Grundlage für die Visualisierungen ist die auf OSGi basierende verteilte Integrationsumgebung *RCE*.

4.3.1 Visualisierung in 2D

Die zweidimensionale Visualisierung basiert auf einer Implementierung mit Webtechnologien im Browser mit Javascript. Die eigentliche Visualisierung wurde mit der Javascript Bibliothek *Data-Driven Documents*[1], kurz D3, umgesetzt. Die Darstellung basiert auf dem *Force-Layout*, welches durch einen kräftebasierten Algorithmus zum Zeichnen von Graphen generiert wurde. Dabei werden Bundles, die tendenziell viele Abhängigkeiten aufweisen, innerhalb einer Punktwolke mittig dargestellt. Bundles mit wenigen Verbindungen sind am äußeren Rand dieser Wolke platziert. Dadurch soll einer Überlappung von Knoten und Kreuzung von Kanten entgegengewirkt werden [31]. Abbildung 4.4 zeigt die Hauptansicht der 2D-Visualisierung im Browser. Jeder Punkt entspricht einem Bundle innerhalb des Softwareprojekts.

Multi-View-Ansatz

Die GUI ermöglicht dem Anwender verschiedene Darstellungsoptionen, die in einer Navigationsleiste rechts neben der eigentlichen Visualisierung zu finden sind. So können entweder im Bezug auf die Darstellung der Bundles oder der Abhängigkeiten Einstellungen vorgenommen werden. Beispielsweise lassen sich ein oder mehrere Bundles durch eine Filterfunktion selektieren. Ebenso kann der Anwender von der Bundle- auf Service- oder Klassensicht wechseln (Abbildung 4.5).

Die View, die Services und Service-Components anzeigt, unterscheidet sich nur geringfügig von der Bundle-View. Die Beziehungen untereinander werden auch hier mithilfe von kräftebasierten Graphen dargestellt. Service-Components werden durch Rechtecke repräsentiert, Services durch Kreise. Service-Components sind nur schwach gekoppelt und stehen in keiner direkten Verbindung zueinander, sondern nur zwischen Service-Components und Services. Auch die Art der Beziehung lässt sich anhand der adjazenten Kante ablesen: das Anbieten eines Services ist durch eine gestrichelte Linie gekennzeichnet, das Benutzen durch eine durchgezogene Linie.

[1] https://d3js.org/

Zusätzlich besteht noch die Möglichkeit, sich die Klassen von einem oder mehreren selektierten Bundles anzeigen zu lassen. Dabei werden sie als farbige Punkte kreisförmig angeordnet. Die Farbe zeigt dabei jeweils an, zu welchem Bundle die Klasse gehört. Die Package-Zuordnung ist ebenfalls durch Abstände zwischen den Kreisen gekennzeichnet. Die Abhängigkeiten zwischen den Klassen sind durch Verbindungslinien visualisiert. In Abbildung 4.5(b) wird durch die Farbgebung der einzelnen Punkte (Klassen) also ersichtlich, dass zuvor in der Bundle-View drei Bundles selektiert wurden (grün, blau, orange). Die Abstände zeigen die jeweilige Anzahl der enthaltenen Packages an. Zusätzlich lassen sich auch noch die Namen der jeweiligen Packages anzeigen.

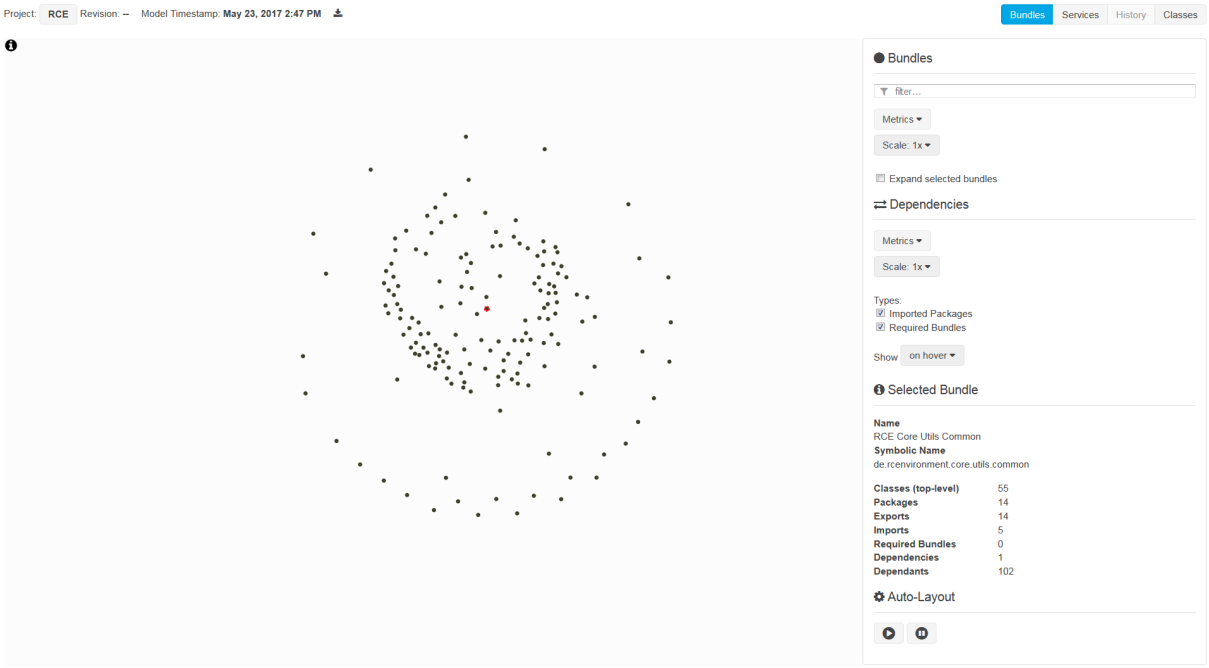
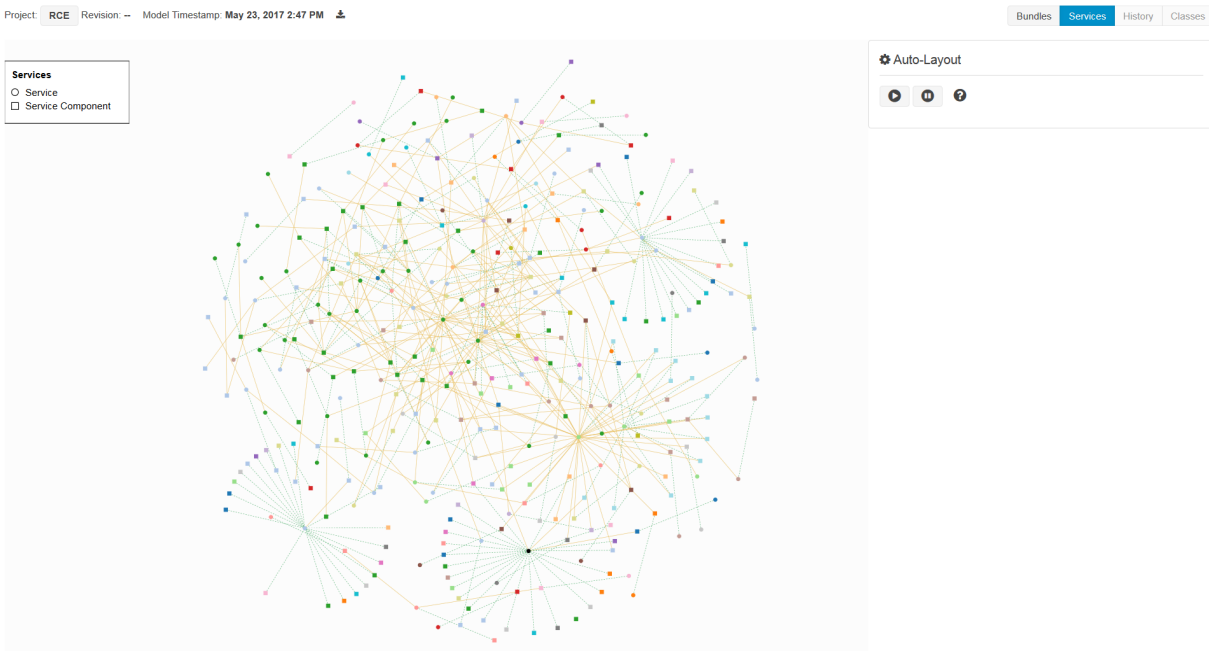


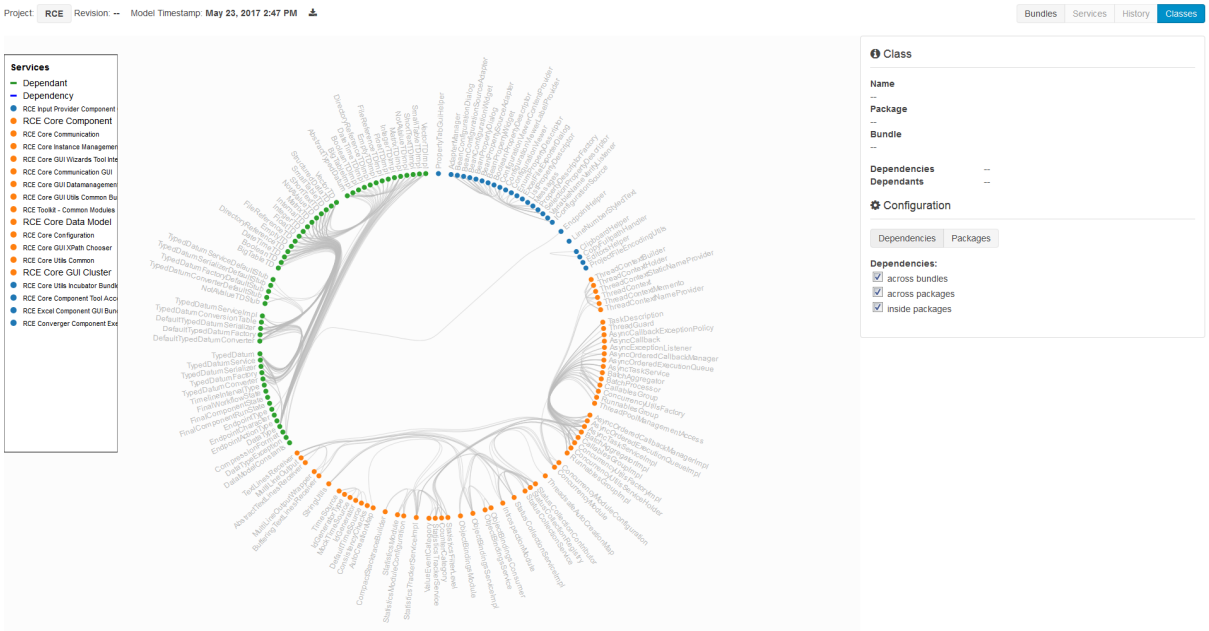
Abbildung 4.4 — Die zentrale Ansicht der Visualisierung auf Bundle-Ebene als Punktwolke.

Darstellungsmodifikationen

In der View auf Bundle-Ebene können verschiedene Metriken eingestellt und somit die Darstellung der einzelnen Softwarekomponenten verändert werden, beispielsweise die Größe der einzelnen Bundles. Je mehr Packages in einem Bundle enthalten sind, desto größer wird das Bundle in der Visualisierung dargestellt. Das gleiche lässt sich für die Anzahl an (top-Level) Klassen und (top-Level) Methoden sowie für Lines of Code anzeigen. Zudem kann entweder noch das Import-Export- oder das Private-Public-Verhältnis innerhalb eines Bundles angezeigt werden (Abbildung 4.6(c)). Dies erfolgt über eine entsprechende Färbung des Kreises. Durch Selektieren eines oder mehrerer Bundles kann man sich zudem die entsprechende Treemap anzeigen lassen, die die Verteilung von Packages und Klassen innerhalb des Bundles veranschaulicht (Abbildung 4.6(b)).



(a)



(b)

Abbildung 4.5 — Die GUI der 2D-Visualisierung erlaubt den Wechsel zwischen mehreren Views: (a) Die Darstellung der Services, ebenfalls dargestellt mithilfe eines kräftebasierten Graphen; (b) Netzdarstellung der Abhängigkeiten auf Klassenebene, die Selektions- und Filtermöglichkeiten erlaubt.

Neben der Möglichkeit, die Bundledarstellung zu verändern, lassen sich auch die Abhängigkeiten zwischen den Bundles auf verschiedene Arten anzeigen (Abbildung 4.6(a)). Zum einen kann der Anwender wählen, welches Bundle welche Packages eines anderen Bundles importiert hat. Außerdem kann man sich anzeigen lassen, welche Bundles für andere Bundles erforderlich sind. Abhängige Bundles sind durch adjazente Kanten verbunden. Durch entsprechende Farbgebung wird noch die Richtung kenntlich gemacht, um die Abhängigkeitsverhältnisse zu verdeutlichen.

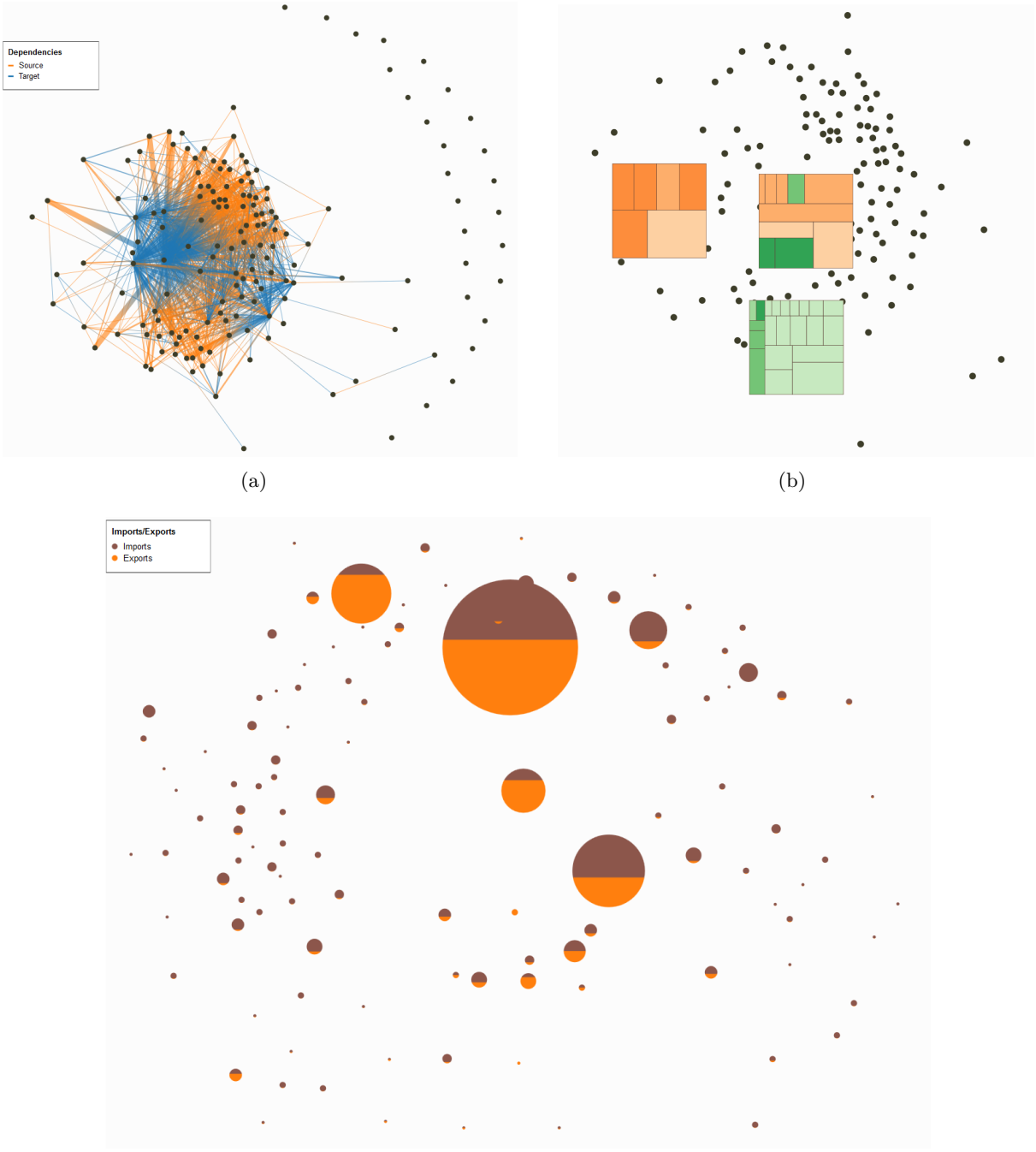
Der letzte Abschnitt der Navigationsleiste zeigt Informationen über ein jeweils ausgewähltes Bundle an, nämlich die Anzahl der

- (Top-Level) Klassen
- Packages
- Exporte
- Importe
- Benötigten Bundles
- Abhängkeiten
- Abhängigen Bundles.

Außerdem werden der Name und der symbolische Name des ausgewählten Bundles angezeigt.

Selektion

Die Selektion eines Bundles erfolgt per Mausklick und dem gleichzeitigen Betätigen der Strg-Taste. Ein selektiertes Bundle wird in der Darstellung durch einen roten Rand gekennzeichnet; parallel dazu werden in der Navigationsleiste die jeweiligen Informationen zu dem Bundle angezeigt.



(a) (b)

(c)

Abbildung 4.6 — Beispiele für verschiedene Darstellungsmodifikationen der 2D-Applikation: (a) Bundles und ihre Abhängigkeiten untereinander; (b) Darstellung der Package-Struktur innerhalb eines Bundles als Treemap. Package und zugehörige Klassen als Rechtecke sind durch die selbe Farbe gekennzeichnet, zusätzlich gibt die Größe einer Klasse deren Anzahl an LOC wieder; (c) Bundles dargestellt nach Größe (Number of Packages) sowie deren Verhältnis von Imports und Exports.

4.3.2 Visualisierung in Virtual Reality

Die Visualisierung in VR basiert auf einer Inselmetapher, entwickelt in *Unity3D*. Dabei wird die gesamte Software auf einem virtuellen Tisch durch einen Ozean mit Inseln dargestellt (Abbildung 4.7), der sich in einem großen Raum befindet. Auch wenn die Softwarevisualisierung vollständig innerhalb des virtuellen Tisches dargestellt ist, so spielt die Umgebung doch keine unerhebliche Rolle für das Gesamtsystem: sie stellt einen festen Rahmen für die Anwendung dar, da sie bekannten VR-Problemen wie Motion Sickness oder Desorientierung entgegenwirkt [40].

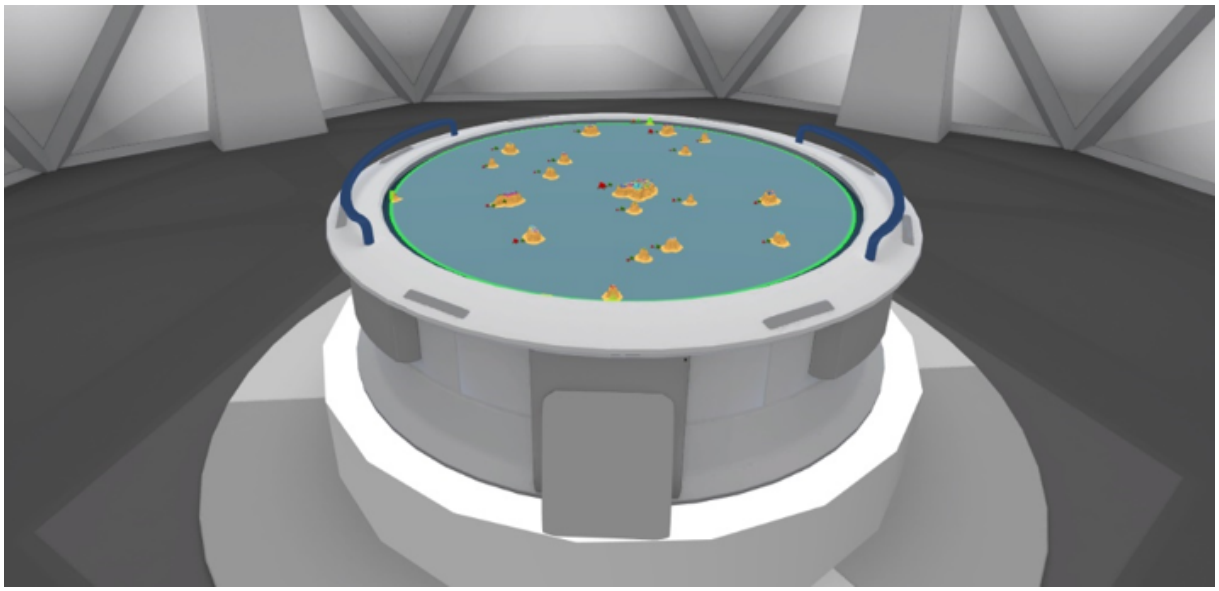


Abbildung 4.7 — Die Ansicht des virtuellen Tisches.

Interaktion und Navigation

Der Nutzer hat die Möglichkeit, mithilfe zweier Controller durch bestimmte Gesten mit der Anwendung zu interagieren. Dabei stehen ihm folgende Arten der Navigation zur Verfügung (Abbildung 4.8):

- Verschieben
- Skalieren
- Rotieren.

So lässt sich beispielsweise in die Anwendung hineinzoomen und die einzelnen Komponenten selektieren. Durch gleichzeitiges Drehen des zweiten Controllers erscheint ein virtuelles Tablet (*Personal Digital Assistant*, kurz PDA), auf dem dann Informationen zu der jeweils ausgewählten Komponente angezeigt werden (Abbildung 4.9). Dazu zählen der Name der ausgewählten Komponente sowie die Anzahl der darin enthaltenen Packages bzw. Klassen.

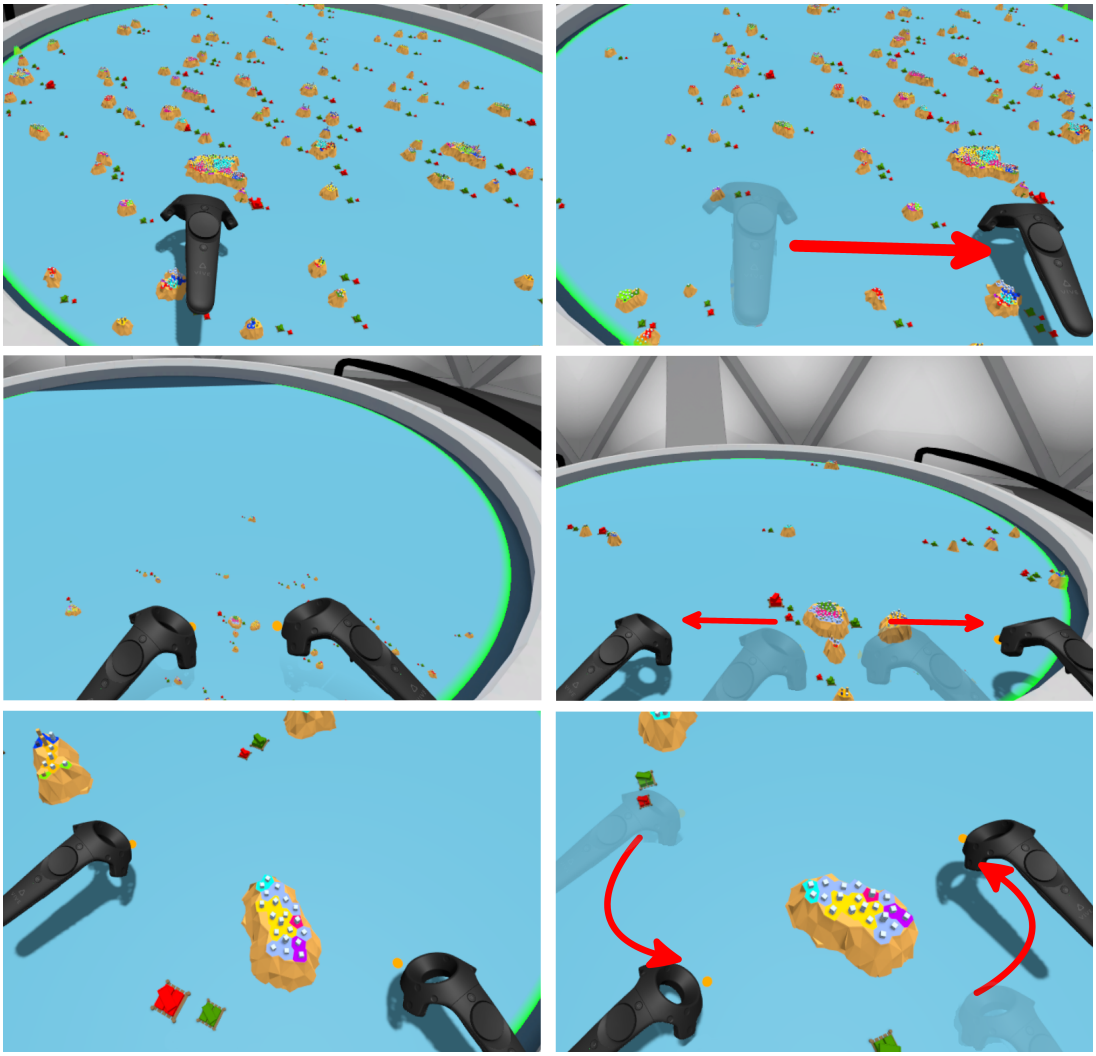


Abbildung 4.8 — Interaktion und Navigation in VR (von oben nach unten): Verschieben, Skalieren und Rotieren.

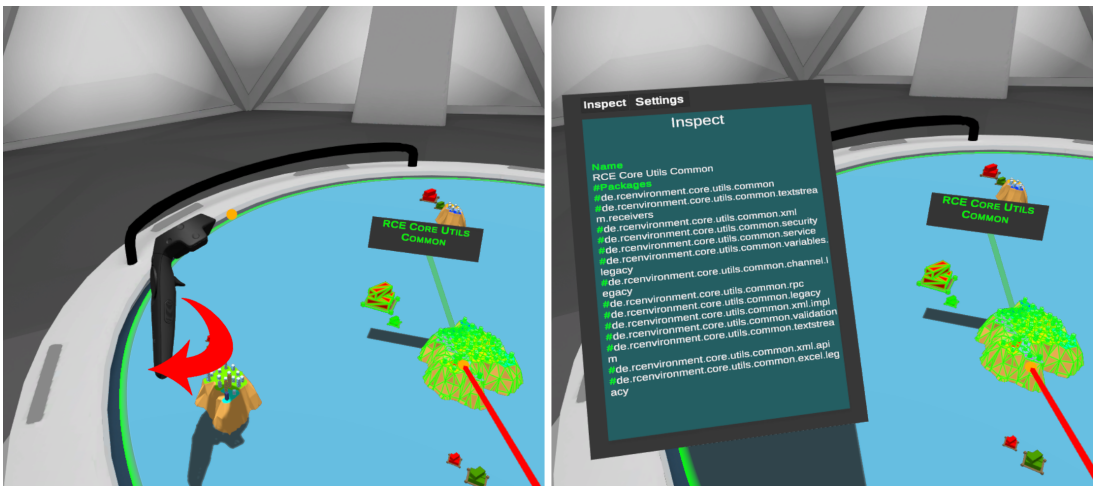


Abbildung 4.9 — Aktivierung des virtuellen Tablets durch Drehung des einen Controllers bei gleichzeitiger Selektion einer Komponente durch den zweiten Controller.

Darstellung der OSGi Komponenten

Abbildung 4.10 zeigt die einzelnen Komponenten der Anwendung im Detail. Jede Insel repräsentiert ein Bundle, welches Regionen (Packages) und darin befindliche Gebäude (Klassen) enthält. Dabei werden alle Komponenten entsprechend ihrer Größe dargestellt. Große Inseln weisen somit eine höhere Anzahl an Packages und Klassen auf als kleinere Inseln. Durch entsprechende Farbgebung der Regionen wird zusätzlich die Abgrenzung zu den anderen Regionen unterstrichen. Die Höhe der Gebäude gibt die Anzahl der Lines of Code der jeweiligen Klasse wieder, das heißt, jedes Stockwerk entspricht n Zeilen Code.

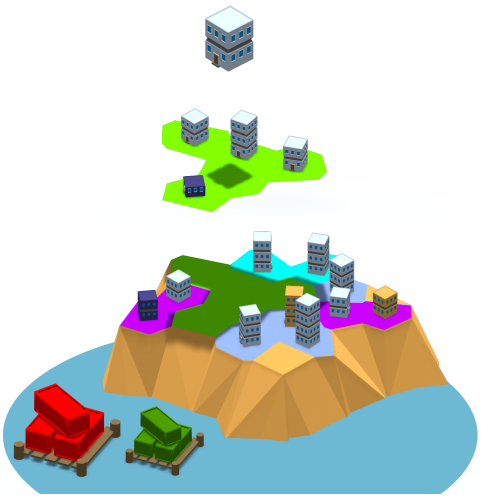
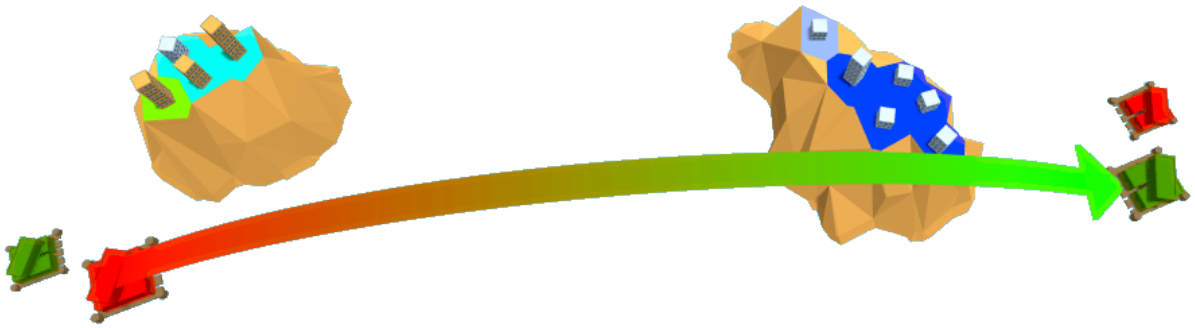
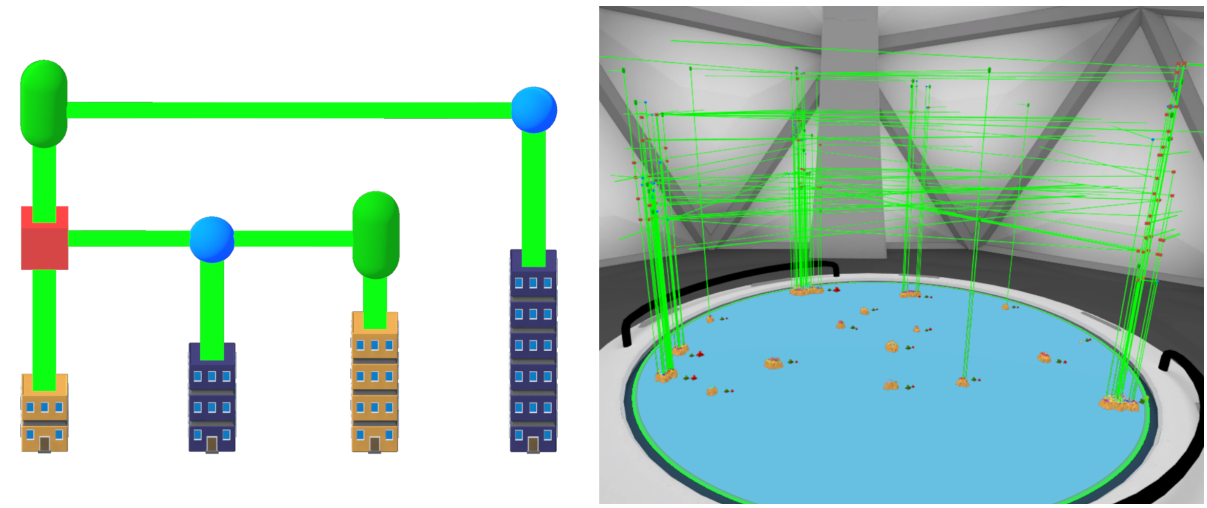


Abbildung 4.10 — Die OSGi-Komponenten der VR-Anwendung als Inselmetapher im Detail: Bundles dargestellt als Inseln (mit Imports und Exports für Package-Abhängigkeiten), Packages dargestellt als Regionen und Klassen dargestellt als Gebäude.



(a)



(b)

Abbildung 4.11 — Abhängigkeiten und Services in VR: (a) Bundles und ihre Abhängigkeiten untereinander; (b) Darstellung der Services und Service Components.

Darstellung der Abhängigkeiten

Darüber hinaus besitzt jede Insel sogenannte *Ports*, die es ermöglichen, sich die Verbindungen dieser Insel zu anderen anzeigen zu lassen, also die Imports und Exports der Packages. Grüne Ports stellen dabei die Imports dar, rote die Exports. Durch Selektion des jeweiligen Ports erscheinen die Abhängigkeiten in Form von Richtungspfeilen (Abbildung 4.11(a)). Alternativ kann sich der Nutzer die Abhängigkeiten aller Bundles untereinander auch über das PDA anzeigen lassen; gleiches gilt für Services und Service-Components (Abbildung 4.11(b)). Eine zusätzliche Funktion bietet hier ferner die Möglichkeit, alle Einstellungen wieder zurückzusetzen.

4.4 Evaluation von Softwarevisualisierungen

In Kapitel 3 wurden bereits verschiedene Evaluationsansätze vorgestellt, die die Usability von Softwaresystemen im Allgemeinen untersuchen. Für den Bereich der Softwarevisualisierungen soll dies nun noch präzisiert werden, um darauf aufbauend ein geeignetes Studiendesign entwickeln zu können. Für die Evaluation von Softwarevisualisierungen eignen sich grundsätzlich Methoden, bei deren Anwendung man zwischen der Erhebung von quantitativen und qualitativen Daten unterscheidet.

Neben Eigenschaften der Visualisierung oder des Algorithmus werden mit quantitativen Methoden auch Merkmale der Interaktion des Anwenders mit dem System gemessen. Typische Fragestellungen, die sich auf die Betrachtung der Visualisierung und des Algorithmus beziehen, wären zum Beispiel, wieviel darzustellende Information auf den Computerbildschirm passt oder wieviel Zeit ein Algorithmus zur Kompilierung der Visualisierung benötigt. Hinsichtlich der Mensch-Computer-Interaktion steht beispielsweise die Frage im Vordergrund, wie lange der Anwender braucht, um auf das System zu reagieren, wieviele Informationen der Darstellung er sich merken kann oder wieviele Aufgaben er innerhalb der Anwendung zu lösen fähig ist.

Die quantitative Erhebung erfordert eine statistische Auswertung von Daten, die innerhalb eines kontrollierten Experiments bzw. einer Studie ermittelt werden. Hierbei werden zwei Arten von Einflussfaktoren berücksichtigt: Zum einen gibt es die unabhängigen Variablen, die von einem Studienleiter kontrolliert und modifiziert werden. Hierzu zählt beispielsweise die Visualisierung ansich. Zum anderen sind die abhängigen Variablen jene Daten, die durch die Studie ermittelt werden. Das kann beispielsweise die benötigte Zeit für das Ausführen einer Aufgabe sein. Nach der Durchführung einer solchen Studie müssen die Daten mithilfe geeigneter statistischer Analyseverfahren entsprechend verarbeitet und interpretiert werden, um daraus Rückschlüsse über die Usability und weitere Design-Implikationen der Anwendung ziehen zu können.

Man unterscheidet zwischen deskriptiver und inferentieller Statistik: Bei deskriptiver Statistik handelt es sich Üblicherweise um statistische Maße wie Mittelwerte, die Aussagen über die zentrale Tendenz eines Datensatzes erlauben, d.h. wie die gemessenen Eigenschaften innerhalb der Stichprobe durchschnittlich verteilt sind. Darüber hinaus können Maße wie die Standardabweichung oder Varianz Aufschluss über die Streuung der Daten geben. Dies ist vor allem bei vielen voneinander abweichenden Werten hilfreich [16]. Die inferentielle Statistik hingegen betrachtet nicht den gesamten Datensatz, sondern vielmehr gewisse Bereiche der Grundgesamtheit. Hierfür werden Analysemethoden verwendet, die beispielsweise zwei Gruppen miteinander vergleichen. Auch die Signifikanz der Unterschiede muss hierbei berücksichtigt werden.

Dem gegenübergestellt liefert die Erhebung qualitativer Daten Informationen über die individuelle, subjektive Wahrnehmung des Anwenders während der Interaktion mit dem System. Dabei eignet sich eine aufgabenbasierte Vorgehensweise, d.h. die Probanden der Studie bekommen Aufgaben vorgelegt, die sie innerhalb der Anwendung nacheinander ausführen sollen. Nach der Ausführung können dann qualitative Daten gewonnen werden, indem der Anwender beispielsweise einen Fragebogen ausfüllt. Eine weitere Methode zur Erhebung qualitativer Daten sind Beobachtungen des Studienleiters, während ein Proband die Aufgaben ausführt [11].

In der vorliegenden Arbeit wurde eine Usability-Studie durchgeführt, die beide Aspekte vereint und somit quantitative und qualitative Daten über die beiden bereits beschriebenen Visualisierungen erhebt. Bevor die Ergebnisse der Studie präsentiert werden, soll im nächsten Kapitel zunächst noch die Methodik und der Ablauf der Studie vorgestellt werden.

5 Konzeption und Durchführung der Studie

5.1 Ziele

Beim Durchführen von Usability-Studien sollte zunächst die Frage im Mittelpunkt stehen, welche Ziele eine solche Studie mit sich bringt. Dabei sollte auch berücksichtigt werden, ob es sich nur um die Bewertung bestimmter Funktionen handelt oder ob das Gesamtsystem evaluiert werden soll. Dabei unterscheidet man zwischen formativen und summativen Usability-Methoden.

Formative Methoden bewerten die Usability eines Systems interativ während des Entwicklungsprozesses. Summative Methoden evaluieren das System im Ganzen, also nach der Implementierung aller gewünschten und benötigten Funktionalitäten. Im nächsten Schritt ist es wichtig, die Nutzergruppe einzuschränken. Dies lässt sich realisieren, indem man nach den Zielen der Nutzer fragt, beispielsweise, ob der Nutzer die Anwendung nur benutzt, um eine bestimmte Aufgabe zu erfüllen, oder ob die Benutzung regelmäßig, zum Beispiel mehrmals am Tag, stattfindet. Dabei ist die Berücksichtigung zweier Hauptaspekte nötig, nämlich die Performance (Effektivität und Effizienz) sowie die Zufriedenheit der Nutzer. Die Performance misst, wie erfolgreich ein Nutzer vorgegebene Tasks lösen kann, also wie hoch die Fehlerrate ist oder wie lange er dabei braucht. Die Zufriedenheit gibt an, wie der Nutzer über die Interaktion mit dem Produkt denkt [55].

Im Falle der vorliegenden Arbeit sollen zwei unterschiedliche Versionen von auf OSGi basierenden Softwarevisualisierungen evaluiert werden, die bereits beide als Gesamtsystem vorliegen, das heißt, es handelt sich um eine summative Evaluierung. Die Softwarevisualisierungen sollen neuen oder unerfahrenen Mitarbeitern, die sich mit bestehenden Softwareprojekten noch nicht allzu gut auskennen, als Hilfsmittel für einen schnellen und übersichtlichen Überblick dienen. Ziel der Applikationen ist also keine regelmäßige Verwendung im täglichen Arbeitsalltag, sondern lediglich als Stütze zum besseren Verständnis von Software.

5.2 Zielgruppe

Neben der Definition der Ziele ist die Auswahl einer passenden Zielgruppe für den Erfolg der Studie mitentscheidend. Im Falle der vorliegenden Arbeit sollen im Folgenden die zwei Zielgruppen vorgestellt werden. Die übergeordnete Zielgruppe von Softwarevisualisierungen ist die Rolle des Softwareentwicklers. Hier existieren verschiedene Use Cases: Zum einen soll die Anwendung einen schnellen Überblick über ein existierendes Softwareprojekt geben. Ein mögliches Beispiel hierfür wäre die Einarbeitung neuer Mitarbeiter in ein bestehendes Projekt. Zum anderen soll ein Entwickler mithilfe der Visualisierung aber auch die Möglichkeit haben, fremde Module schneller an der richtigen Stelle einzuordnen. Dies könnte vorallem in der Weiterentwicklung neuer Module Anwendung finden.

Rolle	Ziel	Nutzen
Entwickler	Neuem Entwickler Überblick über Projekt geben	Einarbeitung neuer Mitarbeiter
	Überblick und Einordnung eines neuen Moduls	Weiterentwicklung fremder Module
	Review der Abhängigkeiten eines Moduls	Einschätzung von Seiteneffekten, Qualitätssicherung, Refactoring
Projektleiter	Ungerichtete Exploration zur Erkennung von Auffälligkeiten	Qualitätssicherung, Projektplanung
	Überblick über Bereiche mit starken Veränderungen	Qualitätssicherung, Projektplanung

Tabelle 5.1 — Verschiedene Use Cases für die beiden Zielgruppen.

Zu guter Letzt sind aber auch gerade die Abhängigkeiten zwischen den Softwaremodulen wichtig, die der Entwickler überprüfen können soll. Dadurch sollen Seiteneffekte auf das Gesamtsystem besser eingeschätzt werden können. Aber auch im Bereich der Qualitätssicherung oder des Refactorings können Softwarevisualisierungen einen Entwickler hier unterstützen.

Neben den Softwareentwicklern sind darüber hinaus Projektleiter, deren informationstechnologische Kenntnisse nicht so stark ausgeprägt sind wie die eines Softwareentwicklers, die zweite mögliche Zielgruppe. Generell sollen Softwarevisualisierungen einem Projektleiter vor allem dahingehend helfen, Auffälligkeiten in der Software erkennen zu können. Aber auch Bereiche mit starken Veränderungen oder sogar Stagnation können in Softwarevisualisierungen deutlicher kenntlich gemacht werden. Letztlich kann einem Projektleiter außerdem ein besserer Überblick über die Wissensverteilung der einzelnen Entwickler über den Code hinweg gegeben werden. All diese Punkte dienen in erster Linie der Qualitätssicherung und der Projektplanung [28].

Im Rahmen der vorliegenden Arbeit beschränkt sich der Fokus auf die erste Zielgruppe der Softwareentwickler mit dem übergeordneten Ziel, sich einen ersten Überblick über das Softwareprojekt zu verschaffen.

5.3 Hypothesen

Auf Basis der bereits vorgestellten Erkenntnisse in den Bereichen Usability und Softwarevisualisierung sollen im Folgenden die Hypothesen präsentiert werden, die durch die innerhalb dieser Arbeit durchgeführte Studie geprüft werden sollen. Die Hypothesen wurden so gewählt, dass sie auf die Hauptaspekte guter Usability per Definition abzielen, nämlich Effektivität, Effizienz und Zufriedenheit. Darüber hinaus basieren sie auf den Qualitätskriterien von Softwarevisualisierungen, die in Kapitel 4.1.3 vorgestellt wurden.

Effektivität

H1 Nutzer lösen die Aufgaben effektiver in 2D als in VR.

H0.1 Nutzer lösen die Aufgaben nicht effektiver in 2D als in VR.

Die Hypothese in Bezug auf die Effektivität der Anwendung liegt darin begründet, dass die Benutzer die Arbeit am Computer gewöhnt sind und sich deshalb vermutlich keine größeren Probleme in der Interaktion ergeben, die einer korrekten Aufgabenlösung im Weg stehen könnten. Außerdem wird davon ausgegangen, dass die graphbasierte Darstellung aufgrund von diversen bereits implementierten Hilfsfunktionen zielführender anzuwenden ist.

Effizienz

H2 Nutzer lösen die Aufgaben schneller in 2D als in VR.

H0.2 Nutzer lösen die Aufgaben nicht schneller in 2D als in VR.

Für die Effizienz, also die jeweils benötigte Zeit zum Lösen einer Aufgabe, gilt eine ähnliche Argumentation. Die Anwendung von VR-Applikationen stellt für die Mehrheit der Nutzer eine neue Erfahrung dar. Außerdem birgt die Visualisierung mithilfe von Metaphern ein höheres Risiko an Ablenkung im Vergleich zur schlichteren, graphbasierten Darstellung. Deshalb wird davon ausgegangen, dass das Lösen der Aufgaben innerhalb der VR-Anwendung mehr Zeit in Anspruch nimmt als in der 2D-Version.

Zufriedenheit

H3 Nutzer lösen die Aufgaben zufriedener in VR als in 2D.

H0.3 Nutzer lösen die Aufgaben nicht zufriedener in VR als in 2D.

Bezüglich der Zufriedenheit wird angenommen, dass aufgrund der durch die VR-Anwendung erzeugte Immersion ein höheres Maß an Exploration und Interaktion erreicht wird. Auch die bildliche Darstellung mithilfe einer Metapher lässt darauf schließen, dass die Nutzer die Applikation als spannender und interessanter wahrnehmen, was sich ebenfalls positiv auf die Zufriedenheit mit dem Gesamtsystem auswirkt.

5.4 Methoden

Das Studiendesign basiert auf der Kombination zweier empirischer Methoden, bestehend aus einem Usability-Test und zwei standardisierten Fragebögen. Bei der Durchführung des Usability-Tests werden durch die Bearbeitung vorgegebener Tasks quantitative Daten gemessen. Dabei liegt der Fokus allein auf die beiden Faktoren Effektivität und Effizienz. Anschließend wird durch den Einsatz der Fragebögen die subjektive Wahrnehmung der Probanden gemessen. Die Daten hierfür liegen zum einen quantitativ durch Skalenbewertung vor, zum anderen qualitativ in Textform durch Kommentarfelder. Mithilfe der Fragebögen können Aussagen über die Zufriedenheit der Nutzer mit dem System getroffen werden. Bevor beide Methoden vorgestellt werden, sollen zunächst noch die Metriken erläutert werden, die zur Erhebung der quantitativen Daten für Effektivität, Effizienz und Zufriedenheit verwendet wurden.

5.4.1 Messmetriken

Task Success

Zur Erhebung der Effektivität wurde *Task Success* gemessen. Diese Metrik gibt an, ob eine Aufgabe korrekt durchgeführt wurde oder nicht. Eine Aufgabe wurde korrekt ausgeführt, wenn er vom Probanden eigenständig, das heißt, ohne jegliche Hilfestellung oder erläuternde Kommentare seitens des Versuchsleiters, ausgeführt wurde.

Eine falsche Aufgabenlösung wurde vermerkt, wenn der Proband die gegebene Aufgabe nicht korrekt lösen konnte; das heißt, er hat entweder eine falsche Antwort gegeben oder das falsche Ergebnis präsentiert. Aufgaben, die abgebrochen oder nur durch Nachfrage beim Versuchsleiter gelöst werden konnten, wurden ebenfalls als falsch bewertet. Für jede korrekt ausgeführte Aufgabe wurde eine 1 notiert, für falsch gelöste Aufgaben eine 0.

Die Ergebnisse der Messung geben die durchschnittliche Erfolgsquote pro Proband und pro Task wieder, angegeben in Prozent.

Time on Task

Time on Task beschreibt die Messung der Zeit, die jeder Studienteilnehmer für das Durchführen einer Aufgabe benötigte. Dabei wurde nicht berücksichtigt, ob die Aufgabe korrekt ausgeführt wurde oder nicht.

Die Erfassung der Zeit erfolgte in Sekunden und startete, nachdem die Aufgabe von dem Probanden laut vorgelesen wurde und endete mit der ausgesprochenen Antwort. Ausnahmen waren Task 4 und 5, bei denen die Zeit angehalten wurde, sobald der Proband die in der jeweiligen Anwendung richtige Darstellung wählte. Je kürzer die Probanden brauchten, um eine Aufgabe zu lösten, desto effizienter wird die Visualisierung bewertet.

Zufriedenheit

Die Erhebung der Zufriedenheit erfolgte über zwei standardisierte Fragebögen, die in Abschnitt 5.4.3 vorgestellt werden. Dabei wurden Werte von eins bis sieben bzw. eins bis fünf gemessen. Die am Ende berechneten Durchschnittswerte geben dann den Grad der Zufriedenheit an, wobei niedrige Werte auf eine eher geringe Zufriedenheit hindeuten; dementsprechend lassen hohe Messwerte auf eine ebenso hohe Zufriedenheit schließen.

5.4.2 Usability-Test

Das Hauptziel der beiden Visualisierungen ist, dass sich die Anwender einen ersten Überblick über den Aufbau des Softwareprojekts verschaffen können. Deshalb war es wichtig, mit den Aufgaben das Erfassen der drei Hauptkomponenten Bundles, Packages und Klassen sowie die Abhängigkeiten und Services abzudecken. Was zudem bei der Entwicklung passender Task Scenarios berücksichtigt werden musste, ist die Vergleichbarkeit beider Systeme. So unterscheiden sich die Visualisierungsformen doch sehr stark in einigen Funktionen. Deshalb mussten die Aufgaben so gewählt werden, dass sie in beiden Visualisierungen durchzuführen sind. Ferner ist darauf zu achten, dass alle Aufgaben einen eindeutigen Start- und Endzustand aufweisen, um die Metriken Task Success und Time on Task zuverlässig erfassen zu können [55]. Nach mehreren Probedurchläufen wurden die folgenden fünf Tasks festgelegt:

Task 1 Selektieren Sie das Bundle mit der größten Anzahl an Packages.
Benennen Sie die Anzahl der Packages.

Task 2 Selektieren Sie ein beliebiges Bundle.
Nennen Sie den Namen des am weitesten entfernten Bundles, zu dem eine Verbindung besteht, und die Anzahl der darin enthaltenen Klassen.

Task 3 Selektieren Sie das Bundle *RCE Core Utils Scripting*.
Benennen Sie die Anzahl der Imports und Exports.

Task 4 Lassen Sie sich die Abhängigkeiten aller Bundles untereinander anzeigen.

Task 5 Lassen Sie sich alle Services bzw. Service Components anzeigen.

Die Reihenfolge der Tasks wurde nach Relevanz der OSGi-Komponenten gewählt. Da die Bundles in beiden Visualisierungen die erste Ebene darstellen, wurde auch mit ihnen in der Aufgabenstellung begonnen. Die folgenden Aufgaben, die die Exploration der Packages und Klassen abdecken, sollen außerdem der Verständnisgewinnung über die Bundlestruktur und den Aufbau von auf OSGi basierenden Softwarearchitekturen beitragen. Die letzten Aufgaben zielen dann auf die Exploration der Abhängigkeiten ab.

5.4.3 Fragebögen

Zur Erhebung subjektiver Daten aus Nutzersicht wurden zwei standardisierte Fragebögen verwendet: Zum einen das *After Scenario Questionnaire*, kurz ASQ, welches direkt nach dem Durchführen eines Szenarienblocks zum Einsatz kommt und auch nur den innerhalb dieses Blocks ausgeführten Task bewertet. Der zweite Fragebogen, die *System Usability Scale*, kurz SUS, wird zur Bewertung des Gesamtsystems am Ende der Aufgaben verwendet.

After Scenario Questionnaire (ASQ)

Beim ASQ handelt es sich um einen von James R. Lewis [23] entwickelten Fragebogen mit drei Items, die als Aussagen über die Aufgabenbearbeitung mit dem System formuliert sind. Der Nutzer soll mithilfe einer siebenstufigen Likert-Skala angeben, inwiefern er den Aussagen zustimmt. Dabei reichen die Items der Skala von 1 ("Stimme überhaupt nicht zu") bis 7 ("Stimme voll zu"). Zur Auswertung der so erhaltenen Daten wird der Durschnittswert berechnet.

Die ersten beiden Aussagen zielen auf die Effektivität und Effizienz des Systems ab. Die Frage nach der Zufriedenheit des Nutzers wird mit allen drei Aussagen beantwortet [55]. Zudem hat der Nutzer die Möglichkeit, zu allen drei Aussagen Bemerkungen mithilfe eines Kommentarfeldes abzugeben.

1. *"Overall, I am satisfied with the ease of comleting the task in this scenario."*
2. *"Overall, I am satisfied with the amount of time it took to complete the task in this scenario."*
3. *"Overall, I am satisfied with the support information (online help, messages, documentation) when completing the task."*

Der Einsatz des ASQ hilft dabei, die durch Beobachtung erhobenen objektiven Daten zu untermauern bzw. zu interpretieren. Darüber hinaus schließt er einen Szenarienblock sauber ab, sodass sich der Nutzer gut auf das neue, folgende Aufgabenszenario einstellen kann.

System Usability Scale (SUS)

Zur subjektiven Bewertung des gesamten Systems wurde die 1986 von John Brooke [6] entwickelte *System Usability Scale* verwendet. Wie auch beim ASQ werden dem Nutzer Aussagen vorgelegt, denen er mithilfe einer diesmal fünfstufigen Skala zustimmt [47]. Dabei reicht die Skala von 1 ("Stimme überhaupt nicht zu") bis 5 ("Stimme voll zu"). Zur Auswahl stehen insgesamt 10 Items - fünf davon sind negativ formuliert, die andere Hälfte positiv.

1. *I think that I would like to use the system.*
2. *I found the system unnecessarily complex.*
3. *I thought the system was easy to use.*
4. *I think that I would need the support of a technical person to be able to use this system.*
5. *I found the various functions in this system were well integrated.*
6. *I thought there was too much inconsistency in this system.*
7. *I would imagine that most people would learn to use this system very quickly.*
8. *I found the system very cumbersome to use.*
9. *I felt very confident using the system.*
10. *I needed to learn a lot of things before I could get going with this system.*

Die Auswertung erfolgt bei der SUS allerdings anders als beim ASQ, bei dem einfach die jeweiligen Mittelwerte ermittelt werden. Bei der SUS kann jedes Item mit Werten zwischen 0 und 4 gewichtet werden. Dabei wird für die Items mit ungeraden Zahlen (1,3,5,7,9), die positiv formuliert sind, jeweils *Wert-1* gerechnet. Für die geraden Items mit negativen Aussagen berechnet man *5-Wert*. Danach werden die Werte addiert und anschließend mit 2.5 multipliziert, um das Gesamtergebnis zu erhalten [55].

Als Maßstab dafür, wie die daraus resultierenden Ergebnisse interpretiert werden können, liefern Bangor et al. (Abbildung 5.1) eine Orientierungshilfe. Demnach gelten Werte unter 50 als nicht akzeptabel, Ergebnisse zwischen 50 und 70 werden als marginal eingestuft und Werte über 70 als akzeptabel.

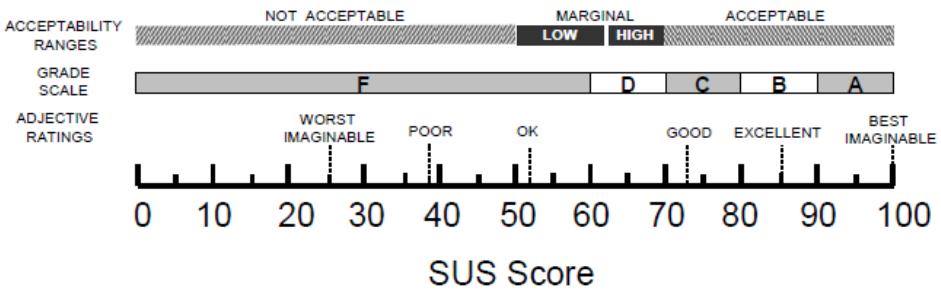


Abbildung 5.1 — Interpretation des SUS Scores [3].

Vorteile der SUS gegenüber anderen standardisierten Fragebögen ist zum einen die hohe Tauglichkeit für Vergleiche zweier Systeme. Zum anderen ist es aufgrund der vergleichsweise geringen Anzahl an Items für den Probanden schnell auszufüllen sowie für den Versuchsleiter schnell auszuwerten, was Zeit spart. Außerdem führt das Berechnen des Scores zu einem präzisen Ergebnis, was leicht verständlich und mithilfe geeigneter Tabellen deutlich interpretiert werden kann.

5.5 Ablauf der Studie

Die Studie wurde im Januar 2019 im Virtual Reality-Labor der Einrichtung Simulations- und Softwaretechnik, Abteilung Intelligente und Verteilte Systeme des Deutschen Zentrums für Luft- und Raumfahrt in Köln Porz durchgeführt. Für die VR-Anwendung stand ein HTC Vive Virtual Reality System mit einer Auflösung von 2160 x 1200 Pixel zur Verfügung, die Evaluation der 2D-Applikation fand an einem 17 Zoll-Laptop der Marke Dell, Intel Core i7 Prozessor, statt. Für die Erstellung der Fragebögen wurde das Online-Tool *SoSci Survey*[1] verwendet.

Die Akquirierung der Studienteilnehmer erfolgte auf persönliche Nachfrage. Als Voraussetzungen galten IT-Hintergrund sowie Programmierkenntnisse in JAVA. Laut Nielsen lassen sich bereits mit 5 Testpersonen 85% der Usabilityprobleme aufdecken [32]. Für die vorliegende Studie wurde eine Stichprobe mit insgesamt 14 Personen gewählt, die per Zufall gleichmäßig auf die beiden Anwendungen verteilt wurden; somit wurden pro Gruppe 7 Personen getestet.

Mit jedem freiwilligen Probanden wurde ein individueller Termin ausgemacht; pro Test Session wurde ein Zeitfenster von 45 Minuten veranschlagt, inklusive Erklärung des Ablaufs der Studie sowie Einführung in das zu evaluierende System. Bei den Probanden, die die VR-Anwendung testeten, entsprach diese Angabe der tatsächlichen durchschnittlichen Bearbeitungszeit, die Teilnehmer der 2D-Gruppe waren durchschnittlich nach etwa 30 Minuten fertig.

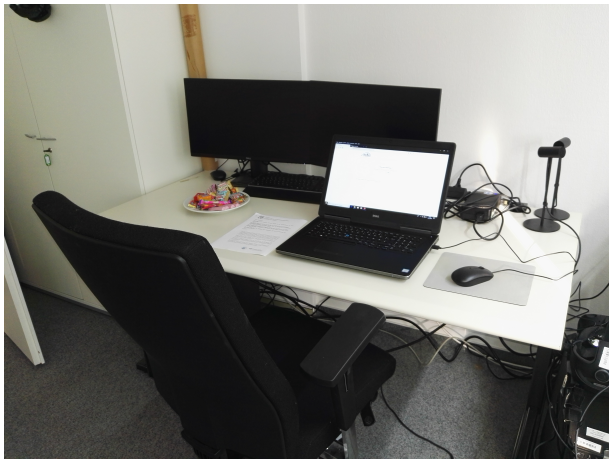
Zu Beginn der Test Session bekam jeder Proband einen Begrüßungstext sowie den Use Case vorgelegt. Nachdem dieser Text gelesen wurde, folgte die Einführung in das System. Dabei wurden der Aufbau sowie die wichtigsten Funktionen und Navigationselemente erklärt. Anschließend erfolgte der eigentliche Usability-Test. Bei der Evaluation der 2D-Applikation wurde der Proband aufgefordert, den ersten Task laut vorzulesen und anschließend mit der Aufgabe zu beginnen. Bei der VR-Anwendung sollte der Proband das Headset aufsetzen und sich in Position bringen. Wenn er das Zeichen gab, bereit zu sein, wurde der erste Task vorgelesen. Der Studienleiter konnte auf einem mit dem VR System verbundenen Laptop mitverfolgen, wie die Aufgabe innerhalb der virtuellen Welt gelöst wurde. Zeitgleich erfolgte bei beiden Anwendungen das Stoppen der Zeit sowie Beobachtungen über Verhalten der Studienteilnehmer.

Sobald der Proband eine Aufgabe gelöst hatte, wurde die Zeit angehalten. Direkt im Anschluss wurde ihm das ASQ vorgelegt, mit dem er die Aufgabe bewerten sollte. Dieser Vorgang wurde bis zur fünften Aufgabe wiederholt (mit Ausnahme von Task 3 in der VR-Anwendung, bei welcher der Proband die Aufgabenstellung mit dem zu findenden Bundle selbst lesen durfte).

Nachdem die letzte Aufgabe durchgeführt und bewertet worden war, folgte noch die Bewertung des Gesamtsystems mithilfe der SUS; außerdem konnte der Proband zusätzliche Kommentare in einem Textfeld hinterlassen.

[1] https://www.soscisurvey.de/

(a)



(b)

(c)

Abbildung 5.2 — Der Versuchsaufbau im Detail: (a) Das VR-Lab der Einrichtung für Simulations- und Softwaretechnik des DLR in Köln; (b) Tisch mit Laptop für das Testen der 2D-Visualisierung; (c) Die einzelnen Bestandteile des HTC Vive Virtual Reality Systems in Verwendung: Headset, Motion Controller und eine von zwei Basisstationen im Hintergrund.

6 Ergebnisse

6.1 Wahl der Analysemethoden

Die im Rahmen der Studie erhobenen Daten liegen sowohl in quantitativer als auch in qualitativer Form vor. Zur Analyse und Interpretation mussten daher geeignete Methoden gefunden werden, die die Ergebnisse möglichst umfassend und strukturiert wiedergeben.

Für die Auswertung der quantitativen Daten wurden ein T-Test bei unabhängigen Stichproben sowie eine mixed ANOVA, die eine einfaktorielle Varianzanalyse mit einer einfaktoriellen Varianzanalyse mit Messwiederholung vereint, durchgeführt. Die beiden Analysemethoden sowie die Interpretation der SPSS-Ausgaben sollen im Folgenden erläutert werden.

Für die Analyse der qualitativen Daten - also die subjektive Meinung der Probanden in verschriftlichter Form, erfasst durch die Kommentarfelder innerhalb des Fragebogens - wurde in Erwägung gezogen, eine quantitative Inhaltsanalyse durchzuführen, die die Häufigkeiten bestimmter Merkmale im Text erfasst. Dabei müssen diese Merkmale in einen bestimmten logischen Kontext zur Beantwortung der Forschungsfragen übertragen werden, was mithilfe eines Kategoriensystems geschieht. Das bedeutet, dass übergeordnete Kategorien gebildet werden müssen, die bestimmte Ausprägungen besitzen. Die Anzahl dieser Ausprägungen wird dann gezählt [44].

Da eine solche zusätzliche Analyse im Rahmen der vorliegenden Masterarbeit allerdings zu umfangreich wäre, wird darauf verzichtet. Stattdessen wurden die innerhalb der Gesamtbewertungen am häufigsten genannten Aspekte zusammengefasst dargestellt. Darüber hinaus dienen die vorliegenden qualitativen Daten als Material, welches die Ergebnisse des quantitativen Datensatzes interpretativ unterstützen soll.

6.1.1 T-Test bei unabhängigen Stichproben

Da im ersten Schritt hinsichtlich der Hypothesenprüfung herausgefunden werden soll, ob im Hinblick auf die Variablen Effektivität, Effizienz und Zufriedenheit signifikante Unterschiede zwischen den beiden Visualisierungen vorliegen, wurde ein T-Test bei unabhängigen Stichproben durchgeführt.

Dieses Analyseverfahren ermöglicht es, auf Basis der in einer Stichprobe ermittelten Kennzahlen Rückschlüsse auf die zugrunde liegende Grundgesamtheit zu ziehen. SPSS liefert hierfür zwei Ausgaben: Einmal *Gruppenstatistiken*, die sowohl Mittelwerte als auch Standardabweichungen der beiden Gruppen angibt. Hier muss zusätzlich das Konfidenzintervall berücksichtigt werden, welches den Wertebereich angibt, in dem der berechnete Mittelwert in der Grundgesamtheit mit einer gewissen Wahrscheinlichkeit liegt. Das Konfidenzintervall ist in der zweiten SPSS-Tabelle

zu finden: *Test bei unabhängigen Stichproben.* Üblicherweise wird hier von einem Prozentwert von 95% ausgegangen. Das Konfidenzintervall gibt sogenannte untere und obere Werte an, die besagen, dass die Grundgesamtheit mit einer Wahrscheinlichkeit von 95% in dem dort angegebenen Bereich liegt.

Darüber hinaus wird der sogenannte Levene-Test in die Berechnung mit eingeschlossen, der die Varianzwerte der beiden Stichproben prüft. Dabei liefert der Levene-Test zwei Varianten: Entweder die Varianzen sind gleich oder sie sind es nicht. Zunächst wird kontrolliert, ob die Varianzen gleich sind. Dies geschieht mithilfe des angegebenen Signifikanzwertes *p*. Liegt dieser unter 0.05, so wird davon ausgegangen, dass die Varianzen nicht gleich sind; der hier unter *Sig. (2seitig)* angegebene p-Wert gibt dann Aufschluss darüber, ob sich die Ergebnisse der beiden Stichproben signifikant unterscheiden; auch hier muss er kleiner als 0.05 sein. Liegt der Wert darüber, so kann davon ausgegangen werden, dass sich die ermittelten Ergebnisse der Stichproben nicht signifikant unterscheiden [7].

6.1.2 Mixed ANOVA

Die Varianzanalyse, kurz *ANOVA*, eignet sich immer dann, wenn untersucht werden soll, wie sich bestimmte Variablen in verschiedenen Gruppen unterscheiden. Eine Sonderform dieser Analysemethode ist die Mixed ANOVA. In der vorliegenden Arbeit soll neben dem Vergleich zwischen den beiden Stichproben zusätzlich herausgefunden werden, welchen Einfluss die ausgeführten Aufgaben jeweils auf die Faktoren Effektivität, Effizienz und Zufriedenheit haben bzw. ob und wie sich die so ermittelten Unterschiede sowohl zwischen den Tasks, als auch zwischen den beiden Fallgruppen unterscheiden. Deshalb wird hier die sogenannte 2 x 5 Mixed ANOVA durchgeführt. Das bedeutet, dass gleichzeitig eine einfaktorielle Varianzanalyse auf die Zwischensubjektfaktoren Gruppe, also 2D und VR, sowie eine einfaktorielle Varianzanalyse mit Messwiederholung auf den Innersubjektfaktor Task (insgesamt 5) gerechnet wird.

Um die Ergebnisse einer solchen Analyse auszuwerten, muss zunächst auf Sphärizität geprüft werden, was mit dem sogenannten Mauchly-Test umgesetzt wird. Der hier erneut unter *Sig.* angegebene p-Wert gibt an, ob Sphärizität gegeben ist, also ob die Unterschiede der Stufen aller unabhängigen Variablen gleich sind. Liegt der Wert bei über 0.05, so wird davon ausgegangen, dass Sphärizität gegeben ist und somit eine Homoskedastizität zwischen den Stufen vorliegt. Ist dies nicht der Fall, können drei verschiedene Korrekturverfahren angewandt werden: Greenhouse-Geisser, Huynh-Feldt und Untergrenze [17]. Dabei gilt Huynh-Feldt als liberalstes, Untergrenze als konservativstes Verfahren [41]. Als verwendetes Verfahren der innerhalb dieser Arbeit durchgeführten Studie wird der Greenhouse-Geiser-Wert angenommen, der in der Mitte der beiden anderen genannten Verfahren liegt.

Im nächsten Schritt erfolgt dann die Bestimmung der Interaktionseffekte. Diese geben darüber Auskunft, ob sich die Tasks in Abhängigkeit von der Gruppe unterscheiden. Dazu liefert SPSS die Ausgabe *Tests der Innersubjekteffekte.* Um die Interaktionseffekte zu prüfen, wird die zweite Zeile betrachtet (Task * Gruppe). Je nachdem, wie das Ergebnis des Mauchly-Tests ausgefallen ist, werden die Ergebnisse entweder aus der Zeile *Sphärizität gegeben* oder *Greenhouse-Geisser* abgelesen. Hier ist wieder der angegebene p-Wert unter *Sig.* relevant - Liegt dieser unter 0.05, so ist ein signifikanter Interaktionseffekt gegeben.

Neben den Interaktionseffekten werden mit der Mixed ANOVA zusätzlich die Haupteffekte der Innersubjektfaktoren ermittelt, die im Falle der vorliegenden Arbeit aussagen, ob sich die Tasks innerhalb der beiden Gruppen signifikant unterscheiden. Dabei wird die Art der Visualisierung nicht berücksichtigt. Hierfür liest man die Ergebnisse aus der ersten Zeile der Tabelle *Tests der Innersubjekteffekte* ab [17].

Im Folgenden werden die Ergebnisse der Stichproben strukturiert nach Effektivität, Effizienz und Zufriedenheit präsentiert. Dabei werden zunächst die Ergebnisse der statistischen Auswertung des T-Tests bezüglich der Unterschiede zwischen den beiden Gruppen präsentiert; Im Anschluss erfolgt die Auswertung der Varianzanalyse mit Betrachtung der Ergebnisse für die einzelnen Tasks.

Die darauffolgende Interpretation schließt dann die Beobachtungen, die vom Studienleiter gemacht wurden, sowie die Fragebögen-Kommentare der Teilnehmer mit ein. Aus den so gewonnenen Erkenntnissen sollen bereits erste Design-Implikationen für die weitere Entwicklung solcher Systeme abgeleitet werden.

6.2 Statistische Auswertung der Daten

Die Auswertung und grafische Darstellung der Daten erfolgte mit IBM SPSS Statistics 25. Die Diagramme über die soziodemographischen Informationen der Studienteilnehmer wurden mit Microsoft Excel erstellt.

Zur Sicherstellung einer möglichst homogenen Teilnehmerstruktur wurde zu Beginn der Evaluation der soziodemographische Hintergrund jedes Probanden abgefragt. Bei der Wahl der Teilnehmer wurde darauf geachtet, dass sie einen informationstechnologischen Hintergrund und zumindest grundlegende Programmierkenntnisse in JAVA aufweisen. Ingesamt nahmen 14 Personen an der Studie teil; Bezüglich der Stichprobencharakteristik lässt sich feststellen, dass 9 und damit 64,3% der Teilnehmer männlich sind. Der größere Teil der Probanden gehört mit ebenfalls 64,3% der Altersgruppe 25-34 an, davon gaben 50% und somit 7 Personen an, als höchsten Bildungsabschluss die allgemeine Hochschulreife vorweisen zu können; die andere Hälfte hatten einen Bachelorabschluss (21,4 %), Masterabschluss (14,3%) oder eine Promotion (14,3%). Darüber hinaus geben 50% der Teilnehmer an, Grundkenntnisse in der Programmierung mit JAVA zu haben. Die anderen 50% bezeichnen ihre Kenntnisse als fortgeschritten. Expertenwissen hatte keiner der Teilnehmer angegeben. Zur Veranschaulichung sind die soziodemographischen Angaben der Studienteilnehmer in Abbildung 6.1 dargestellt.

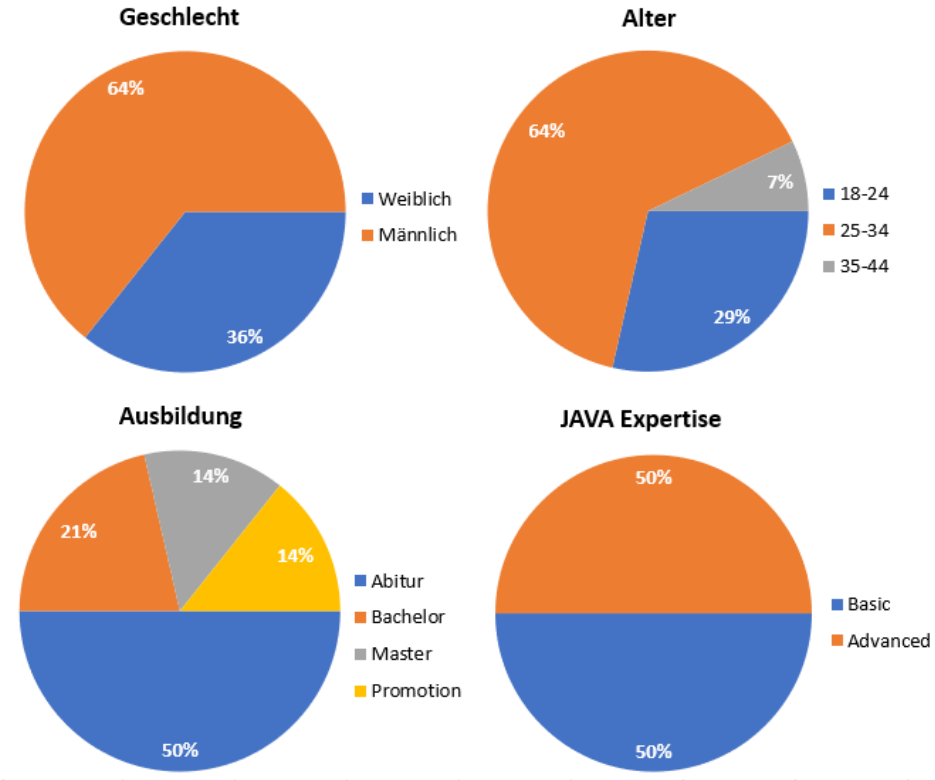


Abbildung 6.1 — Soziodemographische Charakteristika der Studienteilnehmer.

6.2.1 Effektivität

Die Messung des Task Success, also das korrekte Ausführen einer Aufgabe, liefert für die Gruppe der 2D-Probanden insgesamt höhere Mittelwerte (M = 97.14, SD = 7.56) als für die Gruppe der VR-Probanden (M = 82.86, SD = 7.56). Dieser Unterschied konnte mit t(12) = 3.54, p = .004 als signifikant nachgewiesen werden, deshalb kann die Nullhypothese H0.1 verworfen werden und die Alternativhypothese wird bestätigt:

H1 Nutzer lösen die Aufgaben effektiver in 2D als in VR.

Die Ergebnisse im Detail sind in Abbildung 6.2 dargestellt. Aus dieser lässt sich ablesen, welche Prozentwerte die beiden Gruppen bei den einzelnen Tasks erreicht haben. Extreme Unterschiede zeigen sich vor allem bei Task 2: hier lösten 100% der 2D-Anwender die Aufgabe korrekt, während die VR-Anwender nur 57,14% Aufgabenkorrektheit erzielten. Auch Task 1 liefert mit mit 100% korrekt gelösten Aufgaben in der 2D-Gruppe und 85,71% in VR einen deutlichen Unterschied. Auch Task 4 wird von allen 2D-Probanden mit 100% korrekt gelöst, während ein Teilnehmer der VR-Anwendung nicht auf das korrekte Ergebnis kam und die Aufgabe somit nur eine Korrektheit von 85,71% aufweist. Task 3 und Task 5 wurde von beiden mit 100% abgeschlossen.

Die Unterschiede zwischen den gelösten Aufgaben wurden jedoch sowohl zwischen (F(2.81, 33.72) = 1.071, p = .371), als auch innerhalb der Gruppen (F(2.81, 33.72) = .93, p = .433) als nicht signifikant nachgewiesen. Das bedeutet, dass sich die Ergebnisse für die Aufgabenkorrektheit sowohl abhängig als auch unabhängig von der Gruppe nicht signifikant unterscheiden.

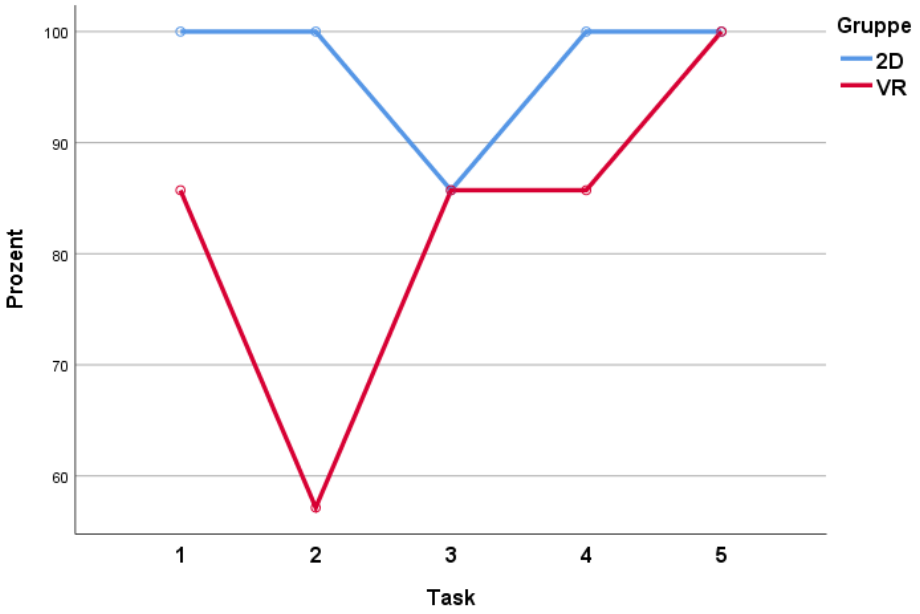


Abbildung 6.2 — Ergebnisse der Auswertung des Task Success hinsichtlich der Effektivität.

6.2.2 Effizienz

Im Hinblick auf die Effizienz, welche durch die benötigte Zeit für das Ausführen der Tasks in Sekunden gemessen wurde, wurden zunächst erneut die jeweiligen Mittelwerte für 2D (M = 212.43, SD = 77.12) und VR (514.00, SD = 175.90) ermittelt. Für diese Werte konnte eine hohe Signifikanz nachgewiesen werden (t(8.23) = -4.15, p = .003).

Deshalb wird auch hier die Nullhypothese H0.2 verworfen und Hypothese 2 wird bestätigt:

H2 Nutzer lösen die Aufgaben effizienter in 2D als in VR.

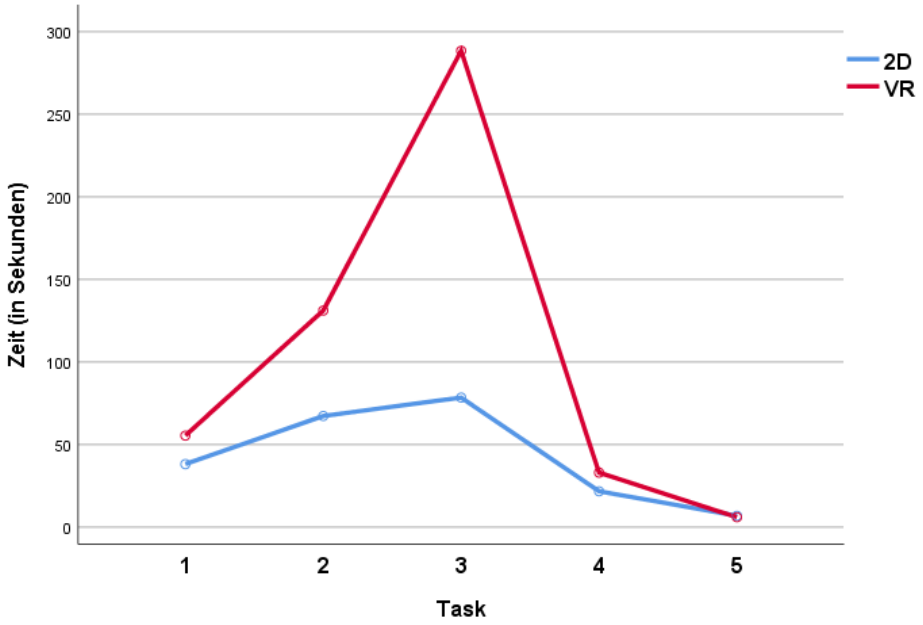


Abbildung 6.3 — Ergebnisse der Auswertung von Time on Task hinsichtlich der Effizienz

Betrachtet man die Ergebnisse im Einzelnen (Abbildung 6.3), so lässt sich ein großer Unterschied zwischen den beiden Gruppen erkennen. Dieser wird vor allem bei Task 3 deutlich: Die Probanden der VR-Gruppe benötigten hier im Mittel 288,43 Sekunden zum Lösen der Aufgabe, während die 2D-Gruppe diese Aufgabe durchschnittlich in nur 78,43 Sekunden löste. Das Durchführen von Task 4 und 5 scheint beiden Gruppen wiederrum keine Schwierigkeiten bereitet zu haben, hier sind die Durchschnittszeiten von 21,71 Sekunden (2D) und 33,00 Sekunden (VR) für Task 4 sowie 6,86 Sekunden (2D) und 6,00 Sekunden (VR) relativ ähnlich. Dies lässt sich auch für Task 1 festhalten, die Probanden der 2D-Gruppe sind mit 38,14 Sekunden im Mittel nicht erheblich schneller als die VR-Gruppe, die durchschnittlich 55,43 Sekunden benötigte. Ein größerer Unterschied sowohl innerhalb als auch zwischen den Gruppen ist wieder bei Task 2 erkennbar. Hier benötigten die Teilnehmer beim Lösen der Aufgabe in 2D knapp eine Minute (67,29 Sekunden), die VR-Gruppe fast doppelt so lang (131,14 Sekunden).

Die Prüfung auf Interaktionseffekte ergab mit $F(2.19, 26.24) = 8.25$, $p = .001$) einen signifikanten Unterschied bezüglich der benötigten Zeit für die einzelnen Tasks zwischen den Gruppen. Für die Unterschiede zwischen den Aufgaben innerhalb der Gruppen lässt sich festhalten, dass auch sie signifikant sind mit $F(2.19, 26.24) = 21.80$, $p < .001$).

6.2.3 Zufriedenheit

Die Zufriedenheit der Probanden mit der jeweiligen Visualisierung wurde sowohl für die einzelnen Tasks als auch für das Gesamtsystem mithife der beiden standardisierten Fragebögen ermittelt. Zunächst werden die Ergebnisse des ASQ präsentiert, im Anschluss folgt die Auswertung der SUS. Für den ASQ werden zusätzlich die Ergebnisse innerhalb der Gruppen präsentiert, um so wieder Unterschiede in der Zufriedenheit in Abhängigkeit der durchgeführten Aufgaben erkennen zu können.

After Scenario Questionnaire (ASQ)

Die Probanden sollten hier nach jeder der fünf durchgeführten Aufgaben drei Items auf einer siebenstufigen Likert-Skala bewerten, wobei 1 "Stimme überhaupt nicht zu" und 7 "Stimme voll zu" entspricht. Für jede Stichprobe wurde der Mittelwert berechnet, wobei als Maximalwert 105 Punkte (pro Aufgabe 21) zu erreichen war. Generell unterscheiden sich die Ergebnisse nicht erheblich, beide Visualisierungen weisen hohe Werte auf, wobei VR ($M = 77.86$, $SD = 14.1$) geringfügig besser bewertet wurde als 2D ($M = 77.14$, $SD = 11.82$). Dieser Unterschied wurde jedoch als nicht signifikant nachgewiesen ($t(12) = -0.10$, $p = .920$).

Insgesamt wird die VR-Anwendung bei Task 1 und 2 sowie bei Task 4 und 5 geringfügig zufriedenstellender wahrgenommen als die 2D-Anwendung. Der höchste zu erreichende Wert ist pro Aufgabe ein Score von 21. Task 1 wird relativ ähnlich bewertet mit einem Score von 16,14 bei 2D und 16,86 bei VR. Innerhalb der Gruppen zeigen sich dann bereits bei der nächsten Aufgabe Unterschiede, da Task 2 von den 2D-Probanden nur noch mit 13,57 Punkten und von den VR-Probanden mit 14,86 Punkten bewertet wird. Task 3 weist in der VR-Anwendung dann einen starken Unterschied im Vergleich zu den anderen Aufgaben auf: Die Teilnehmer dieser Gruppe bewerten die Aufgabe nur mit 8,86 Punkten, während die Zufriedenheit in der 2D-Gruppe mit 14,00 Punkten recht konstant bleibt. Task 4 wird dann wieder von beiden Gruppen deutlich positiver bewertet: 2D erreicht einen Score von 16,71, VR einen Score von 18,43. Ähnlich Ergebnisse liefert Task 5, wenngleich VR noch einmal deutlich besser bewertet wird mit einem Score von 18,86 Punkten. Dies entspricht gleichzeitig auch der Aufgabe, die das höchste Maß an Zufriedenheit in der VR-Anwendung aufweist. Die 2D-Anwendung erreicht einen Score von 16,71 Punkten und wird damit als gleichermaßen zufriedenstellend wahrgenommen wie Task 4.

Bei der Auswertung der Ergebnisse für die einzelnen Aufgaben konnte ein statistisch signifikanter Effekt von Task auf die Zufriedenheit mit $F(2.73, 32.73) = 9.62$, $p < .001$ nachgewiesen werden. Und auch bezüglich der Interaktionseffekte lässt sich festhalten, dass es eine statistisch signifikante Interaktion von Task und Gruppe gibt ($F(2.73, 32.73) = 3.01$, $p = .048$).

Abbildung 6.4 stellt die Ergebnisse der beiden Analysen anschaulich dar.

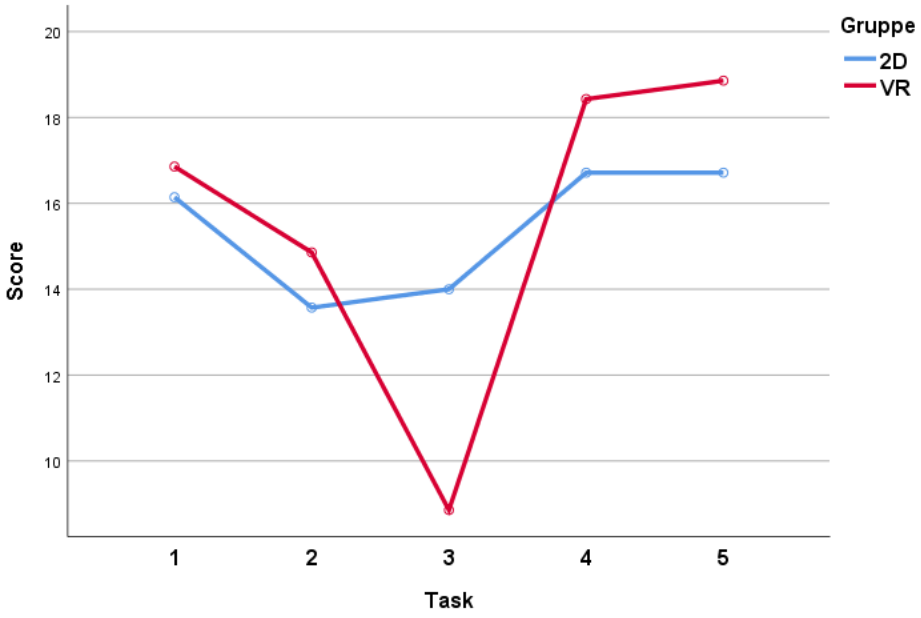


Abbildung 6.4 — Ergebnisse der Auswertung des ASQ hinsichtlich der Zufriedenheit.

System Usability Scale (SUS)

Die SUS basiert auf einer fünfstufigen Bewertungsskala mit 10 zu bewertenden Items. Hier reicht die Skala von 1 "Stimme überhaupt nicht zu" bis 5 "Stimme voll zu". Anders als beim ASQ werden hier nicht einfach die Werte für die einzelnen Items addiert, sondern durch ein bestimmtes Verfahren der Score berechnet, wobei ein Wert von maximal 100 erreicht werden kann. Insgesamt weist 2D (M = 54.29, SD = 9.97) bessere Werte auf als VR (M = 49.64, SD = 8.22). Vergleicht man die Ergebnisse mit der von Bangor et al. [3] entwickelten Tabelle zur Bewertung des SUS-Scores (siehe Kapitel 5.4.3), so müssen beide Ergebnisse als nicht akzeptabel bzw. grenzwertig interpretiert werden. Die Unterschiede zwischen den beiden Gruppen wurde allerdings als nicht signifikant nachgewiesen ($t(12) = 0.95$, $p = .361$).

Da auch die Unterschiede der Ergebnisse des ASQ nicht signifikant sind, wird H3 wird verworfen und die Nullhypothese bestätigt:

H0.3 Nutzer lösen die Aufgaben nicht zufriedener in VR als in 2D.

6.3 Interpretation

In diesem Kapitel sollen nun die bereits dargestellten Ergebnisse bezüglich der Tasks im Detail interpretiert werden, um so die Schwachstellen der Systeme aufzudecken und gegebenenfalls Schlussfolgerungen für die Weiterentwicklung zu ziehen. Für einen besseren Überblick sind die bisherigen Ergebnisse in Tabelle 6.1 noch einmal zusammengefasst und gerundet dargestellt.

	2D	**VR**
Effektivität	97% korrekt gelöste Aufgaben	83% korrekt gelöste Aufgaben
Effizienz	∅212 Sekunden	∅514 Sekunden
Zufriedenheit	**ASQ**: $\bar{x} = 77$ Punkte **SUS**: $\bar{x} = 54\%$	**ASQ**: $\bar{x} = 78$ Punkte **SUS**: $\bar{x} = 50\%$

Tabelle 6.1 — Ergebnisse der Durchschnittswerte für Effektivität, Effizienz und Zufriedenheit.

Insgesamt erzielt die 2D-Visualisierung in fast allen Fällen bessere Ergebnisse als die VR-Visualisierung. Dabei sind die Ergebnisse der Messungen bezüglich Effektivität und Effizienz signifikant, bei der Zufriedenheit konnte hingehen kein signifikanter Unterschied festgestellt werden.

Im Folgenden sollen nun die einzelnen Tasks im Detail betrachtet werden, was auch die vom Studienleiter gemachten Beobachtungen sowie die Kommentare und Bewertungen der Studienteilnehmer miteinschließt.

Am Ende erfolgt eine Gegenüberstellung der Ergebnisse, die zusätzlich die Gesamtbewertung der Probanden über die beiden Systeme miteinbezieht. Diese wurden ebenfalls über den Fragebogen erfasst.

Task 1

Selektieren Sie das Bundle mit der größten Anzahl an Packages.
Benennen Sie die Anzahl der Packages.

Effektivität: Alle Probanden der 2D-Anwendung lösten diese Aufgabe korrekt, bei der VR-Anwendung kam eine Person auf das falsche Ergebnis. Dieser Teilnehmer selektierte das falsche Bundle. Er erkannte zwar, dass die größte Insel auch dem Bundle mit der größten Anzahl an Packages entspricht, nahm sich jedoch nicht genug Zeit, um durch die Anwendung zu navigieren und wählte somit das Bundle aus, von dem er dachte, dass es das größte sei. Ein Teilnehmer der 2D-Anwendung löste zwar die Aufgabe korrekt, hatte aber zunächst noch Probleme, das Bundle zu selektieren (durch Strg + Mausklick), was sich letztlich jedoch nicht auf seine Zeit auswirkte.

Effizienz: Die Probanden der 2D-Anwendung benötigten zum Lösen von Task 1 durchschnittlich 38,14 Sekunden, in VR lag die durchschnittliche Zeit bei 55,43 Sekunden. Ein Studienteilnehmer war mit 15 Sekunden verhältnismäßig schnell. Die Versuchsperson, die in der VR-Anwendung als einzige die Aufgabe nicht korrekt lösen konnte, war mit 18 Sekunden verhältnismäßig schnell, was eine Erklärung für das falsche Ergebnis sein könnte. Ein 2D-Anwender brauchte mit 64 Sekunden fast doppelt so lange wie der Durchschnitt; hier wurde beobachtet, dass der Proband nervös war und sich durch das Stoppen der Zeit unter Druck gesetzt fühlte. Ein weiterer Teilnehmer hatte zunächst Schwierigkeiten, in der 2D-Anwendung den Unterschied zwischen Bundles und Packages zu erkennen, weshalb er auch mit 71 Sekunden von allen Probanden der 2D-Anwendung die Person war, die die meiste Zeit zum Lösen der Aufgabe benötigte.

Zufriedenheit: In Bezug auf die Zufriedenheit finden sich keine signifikanten Unterschiede zwischen den beiden Visualisierungen bei Task 1. Die 2D-Anwendung wurde durchschnittlich mit 16,14 Punkten gewertet, bei VR sind es 16,86. Ein Anwender der 2D-Visualisierung kritisierte das Schriftbild der Menüleiste: Es sei "zu einheitlich, keine Hervorhebungen der unterschiedlichen Optionen. Alles sieht gleich aus". Ein anderer Teilnehmer in der 2D-Gruppe gab an, dass es verwirrend sei, zusätzlich die Strg-Taste zu drücken, um ein Bundle zu selektieren und somit zusätzliche Informationen zu erhalten ("Would expect that clicking on it is enough"). Zwei Probanden in der VR-Gruppe erwähnten, dass zusätzliche Hilfe zum Lösen der Aufgabe nicht nötig war. Es wird angenommen, dass sich diese Kommentare auf die Bewertung des dritten Items beziehen (*"Overall, I am satisfied with the support information (online help, messages, documentation) when completing the task"*). Ein anderer Studienteilnehmer in der VR-Gruppe lobte die Darstellung des Größenunterschiedes der Inseln: "Die Aufgabe war in diesem Fall (für mich) nicht schwierig zu lösen, da einige Inseln durch die Größe herausgestochen haben."

Design-Implikation: Generell schien das Lösen von Task 1 für den Großteil der Probanden kein Problem darzustellen. Lediglich ein Teilnehmer der VR-Anwendung kam auf das falsche Ergebnis, da er nicht das größte Bundle wählte. Hier könnte man die Größenunterschiede vielleicht noch auf allen Ebenen deutlicher hervorheben, denn gerade, wenn man noch nicht weit genug in die Anwendung hineingezoomt hat, erscheint eine konkrete Differenzierung schwierig. Ein Ansatz wäre, die größte Insel (die somit auch das Hauptbundle darstellt) eindeutiger zu kennzeichnen.

Um einem solchen Problem in der VR-Anwendung zukünftig entgegenzuwirken, sollte auf ein noch stärkeres Herausarbeiten der Größenunterschiede zwischen den Inseln geachtet werden. Dies könnte mithilfe zusätzlicher Metriken geschehen, beispielsweise durch entsprechende Farbgebung oder ein einprägsames Symbol neben der Insel. Gerade bei geringfügigen Unterschieden (beispielsweise Bundles mit 42 und 35 Packages) ist es für den Anwender auf den ersten Blick nicht ersichtlich, bei welcher Insel es sich um die größte handelt. Hier könnte auch das Einbeziehen weiterer Metriken helfen, zum Beispiel durch Farbe.

Hinsichtlich der 2D-Anwendung könnte die Gestaltung der Menüleiste noch verbessert werden, da hier eine zu starke Einheitlichkeit kritisiert wurde. Die in Kapitel 3.3 vorgestellten Gestaltgesetze könnten hier Unterstützung liefern, beispielsweise durch noch deutlichere Hervorhebung der Hilfsfunktionen. Da in der 2D-Anwendung das Nutzen des Drop Down-Menüs zum Anzeigen der Größe der einzelnen Bundles unumgänglich für das Lösen dieser Aufgabe war, könnte der Anwender auch hier durch zusätzliche Metriken unterstützt werden. In Bezug auf das Drop Down-Menü wäre beispielsweise eine zusätzliche Beschreibung oder Überschrift hilfreich.

Task 2

Selektieren Sie ein beliebiges Bundle.
Nennen Sie den Namen des am weitesten entfernten Bundles, zu dem eine Verbindung besteht, und die Anzahl der darin enthaltenen Klassen.

Effektivität: Das Durchführen von Task 2 liefert einen hoch signifikanten Unterschied zwischen den beiden Anwendungen. Während hier wieder alle Probanden der 2D-Gruppe die Aufgabe korrekt ausführen, liegt die Fehlerquote in der VR-Gruppe bei 42,86%, was drei von sieben nicht korrekt ausgeführten Aufgaben entspricht. Die Beobachtungen zu den betroffenen Probanden zeigen, dass die fehlerhafte Ausführung auf mangelnden Kenntnissen über die Interaktion mit der Anwendung sowie einem Fehlinterpretieren der Aufgabenstellung beruht. Zwei der betroffenen Probanden wählten beispielsweise nicht die Abhängigkeitsdarstellung über die Ports, sondern ließen sich direkt alle Abhängigkeiten über das PDA anzeigen, was zu einer unübersichtlichen und verwirrenden Darstellung führte. Darüber hinaus wählten beide Probanden kein beliebiges Bundle; einer der beiden ging davon aus, das Hauptbundle selektieren zu müssen. Zusätzlich hatte diese Person Schwierigkeiten, die Anzahl der enthaltenen Klassen herauszufinden. In der VR-Anwendung lässt sich dies nur durch Zählen der auf der Insel befindlichen Gebäude oder durch Selektion der einzelnen Regionen und dem gleichzeitigen Anzeigen auf dem PDA ermitteln. Die betroffene Person versuchte jedoch, die Anzahl über Selektion der Insel herauszufinden, was dazu führte, dass die falsche Antwort - nämlich die Anzahl der enthaltenen Packages - genannt wurde. Der andere Proband ließ sich ebenfalls durch die verwirrende Darstellung aller Abhängigkeiten der Bundles untereinander ablenken und wählte letztendlich das am weitesten aussen gelegene Bundle. Die Frage nach der Anzahl an enthaltenen Klassen beantwortete dieser Proband aber richtig. Der dritte Teilnehmer, der Task 2 nicht korrekt ausführte, wählte lediglich ein beliebiges Bundle und nannte die Klassenanzahl von diesem. Hier wird davon ausgegangen, dass die Aufmerksamkeit des Probanden beeinträchtigt war und die Frage nicht vollständig aufgenommen wurde. Ein anderer Teilnehmer der VR-Gruppe nannte zunächst nur die Anzahl an Packages, bemerkte dann aber von selbst, dass die Anzahl der Klassen gefragt war und korrigierte sein Ergebnis.

Effizienz: Die Probanden der VR-Anwendung brauchten zum Lösen von Task 2 mit durchschnittlich 131,14 Sekunden deutlich länger als die 2D-Gruppe (67,29 Sekunden). Hier lässt sich mit der bereits erwähnten Darstellung der Abhängigkeiten argumentieren, die in der VR-Anwendung schnell unübersichtlich und verwirrend wird. Die Darstellung der Abhängigkeiten über die Ports eines selektierten Bundles scheint für viele Probanden nicht intuitiv, weshalb häufig versucht wurde, das PDA zur Hilfe zu nehmen. Dies erlaubt aber nur die Darstellung aller Abhängigkeiten. Darüber hinaus ist auch das Zählen der enthaltenen Klassen als sehr zeitaufwändig einzustufen. Die 2D-Anwendung erleichtert dies, indem hier alle wichtigen Informationen zu einem selektierten Bundle in der Menüleiste stehen. Hier hatte jedoch ein Teilnehmer zunächst Schwierigkeiten, die Darstellung der Abhängigkeiten über das Drop Down-Menü zu aktivieren. Dieser Teilnehmer benötigte mit 102 Sekunden überdurchschnittlich lange, um die Aufgabe zu lösen

Zufriedenheit: Die 2D-Gruppe bewertete Task 2 mit 13,57, die VR-Gruppe mit 14,86 Punkten, was beides einer mittelmäßigen Zufriedenheit entspricht. Beide Gruppen bewerteten diese Aufgabe schlechter als den vorherigen Task 1. Bei der 2D-Anwendung wurde die fehlende Angabe dafür, welches das am weitesten entfernte Bundle ist, kritisiert ("Das am weitesten entfernte Bundle musste ich nach Augenmaß bestimmen"). Ein Teilnehmer der VR-Gruppe erwähnte die fehlende Anzeige der Klassenanzahl auf dem PDA ("The total number of classes in a bundle could already be shown on the virtual tablet").

Design-Implikation: Die aus den Beobachtungen und Kommentaren resultierenden Erkenntnisse liefen diverse Ansätze für die Weiterentwicklung der beiden Anwendungen. So finden sich signifikante Unterschiede zwischen den Gruppen bezüglich der Effektivität und Effizienz; in beiden Bereichen scheint die VR-Anwendung größere Defizite aufzuweisen als die Darstellung in 2D. Das liegt zum einen daran, dass die 2D-Anwendung eine kompaktere Ansicht aller nötigen Informationen zu einem Bundle liefert. Dies ist in VR nicht der Fall, da hier relativ umständlich die einzelnen Komponenten selektiert werden müssen, um an die jeweils erwünschte Information zu gelangen. Dies ließe sich jedoch ebenfalls in der VR-Anwendung realisieren durch das Anzeigen aller Informationen auf dem PDA bei gleichzeitiger Selektion des Bundles.

Darüber hinaus ist die Ansicht aller Abhängigkeiten über das PDA fragwürdig. Zwar lässt sich durch das Anzeigen ebendieser ein Eindruck über den Umfang eines Softwareprojekts gewinnen, wirklich hilfreich für den Entwickler ist dies aber nicht. Hilfreicher wäre, ebenfalls über das PDA bei wiederrum gleichzeitiger Selektion eines Bundles die Abhängigkeiten sichtbar zu machen und nicht umständlich über die Ports. Ein weiterer Nachteil der Ports ist das ständige Hinein- und Hinauszoomen, das einerseits nötig ist, um die Ports zu selektieren, andererseits, um zu erkennen, zu welchen anderen Bundles die Abhängigkeiten bestehen.

Auch die 2D-Anwendung bereitete gewisse Schwierigkeiten bei der Darstellung der Abhängigkeiten, die ziemlich versteckt über ein Drop Down-Menü in der Informationsleiste zu finden ist. Wie schon bei Task 1 könnte entweder die deutlichere Kennzeichnung des Drop Down-Menüs Abhilfe verschaffen, beispielsweise durch eine separate Überschrift. Eine andere Alternative wäre, das Anzeigen der Abhängigkeiten neben anderen Funktionen direkt beim Selektieren des entsprechenden Bundles anzubieten, beispielsweise durch ein Drop Down-Menü, welches durch Rechtsklick eingeblendet wird.

Task 3

Selektieren Sie das Bundle "RCE Core Utils Scripting".
Benennen Sie die Anzahl der Imports und Exports.

Effektivität: Sowohl in der 2D- als auch in der VR-Gruppe gab es jeweils einen Probanden, der Task 3 nicht korrekt ausführte. Daraus ergibt sich eine Erfolgsquote von 85,41% in beiden Anwendungen. Der betroffene Proband der 2D-Gruppe nannte zwar am Ende die richtige Anzahl an Imports und Exports, dies erfolgte allerdings nur durch Hilfe des Studienleiters; der Proband wollte das gewünschte Bundle mithilfe der Filterfunktion finden, da er aber nicht die richtige Schreibweise verwendete, wechselte er die View. Auf Nachfrage ging er dann wieder zurück auf die Bundle-View und kam dann doch mithilfe der Filterfunktion und mehreren unterschiedlichen

Versuchen bezüglich der Schreibweise auf das richtige Bundle. Die Anzahl der Imports und Exports las er korrekt und ohne Hilfe von der Informationsleiste ab. Weitere Beobachtungen zeigen, dass andere Probanden ebenfalls zunächst Probleme mit der abweichenden Schreibweise in der Filtermaske hatten ("Suchfunktion kann nur exakte Bezeichung finden, nicht die Schlagwörter", "It is not stated which features the filter provides and what I can search for"), was die korrekte Ausführung der Aufgabe aber letztlich nicht beeinträchtigte.

Der betroffene Proband in der VR-Gruppe versucht zunächst, das Bundle mithilfe der Abhängigkeitsverbindungen zu finden. Da die VR-Anwendung keine Filterfunktion anbietet, muss das gewünschte Bundle durch Selektieren der einzelnen Inseln gefunden werden. Der Proband benötigte dafür sehr viel Zeit und gab schließlich auf. Ein anderer Teilnehmer versuchte, über die Exportverbindungen ähnlich klingender Bundles, deren Namen ebenfalls mit "RCE Core Utils" beginnen, die gesuchte Insel zu finden, was ihm allerdings nicht half: "Finding islands named alike did not result in a faster search, because the island to be found was far off and I expected it to be closer. I revisited some islands which was ineffective."

Effizienz: Die beiden Anwendungen unterscheiden sich hoch signifikant bezüglich der benötigten Zeit für das Ausführen von Task 3. Die 2D-Gruppe benötigte hier durchschnittlich 78,43 Sekunden, die VR-Gruppe 288,43 Sekunden. Als wesentlicher Grund ist hierfür die fehlende Filterfunktion der VR-Anwendung zu nennen. Die Inseln müssen einzeln selektiert und deren Namen auf dem PDA abgelesen werden. Auch die Anzahl der Imports und Exports kann hier nicht einfach ermittelt werden; sie müssen zunächst durch Selektion der Ports "aktiviert" und anschließend einzeln gezählt werden. Obwohl für diese Aufgabe ein Bundle mit einer geringen Anzahl an Imports und Exports gewählt wurde und somit das Zählen vereinfachte, ist die Aktivierung der Abhängigkeitsdarstellung bereits sehr zeitintensiv. Darüber hinaus mussten sich die Probanden bereits beim Absuchen der einzelnen Inseln stark konzentrieren, was dann das Zählen der Imports und Exports zusätzlich erschwerte.

Zufriedenheit: Mit 8,86 Punkten bewerteten die Probanden der VR-Gruppe diese Aufgabe am schlechtesten. Auch die 2D-Gruppe empfand diese Aufgabe als am wenigsten zufriedenstellend, obgleich hier mit 14,00 Punkten ein deutlich höherer Score erzielt wurde als in der VR-Anwendung. Die Beobachtungen zeigen hier viele Anzeichen von Frustration: "Ist das ätzend" und "Warum steht hier keine Zahl?" Dies schlägt sich auch in den Kommentaren der Anwender nieder. So bewertete ein Proband die Aufgabe als "quite annoying and confusing to check each single island; number of imports and exports had to be counted which is confusing as well". Ein anderer Proband bewertete die Aufgabe als "very simple task that takes too much effort". Ein weiterer Kommentar bezieht sich auf das Zählen der Imports und Exports: "Es wäre besser, wenn die Information auf dem Tablet stehen würde." Die 2D-Gruppe kommentierte überwiegend die Suchfunktion; die Kommentare beziehen sich hier kritisch auf die abweichende Schreibweise in der Suchmaske, welche bereits erwähnt wurden.

Design-Implikation: Die Ergebnisse legen eindeutig nahe, dass die Implementierung einer Suchfunktion in der VR-Anwendung sehr sinnvoll wäre, um dem Anwender das Auffinden eines spezifischen Bundles zu erleichtern. Dies wäre in Bezug auf alle drei Faktoren Effektivität, Effizienz und Zufriedenheit gewinnbringend, da dadurch die Aufgabenkorrektheit und Zeiteinsparung erhöht und somit eine stärkere Zufriedenheit erzielt werden könnte; dies wird auch durch die besseren Werte der 2D-Gruppe deutlich. Allerdings gibt es auch hier Verbesserungsbedarf, da die Suchmaske als Eingabe nur den symbolischen Namen des Bundles

(de.rcenvironment.core.utils.scripting) erkennt. Entweder muss dies in der Informationsleiste als Hinweis angegeben sein oder die Suchmaske muss so modifiziert werden, dass sämtliche Schreibweisen genutzt werden können. Denkbar wäre auch die Implementierung eine Suchmaske, die die Eingabe von regulären Ausdrücken erlaubt - so könnten darüber hinaus mehrere Bundles gleichzeitig gefunden werden.

Und auch das Anzeigen von wichtiger Information - in diesem Fall die Anzahl der Imports und Exports - auf dem PDA würde erneut eine hilfreiche Funktion für den Anwender darstellen, da dies bisher nur sehr umständlich durch Zählen der Verbindungen über die Ports möglich ist.

Ein Proband in der VR-Gruppe merkte an, dass die Kennzeichnung von bereits "besuchten" Inseln bei dieser Aufgabe möglicherweise hilfreich gewesen wäre. Hier wäre eine Tracking-Visualisierung mit Graphen denkbar, die über das PDA aktiviert werden könnte, ähnlich der Darstellung der Abhängigkeiten. Eine weitere Alternative wäre die Darstellung bereits besuchter Inseln mithilfe einer Liste auf dem PDA.

Task 4

Lassen Sie sich die Abhängigkeiten aller Bundles untereinander anzeigen.

Effektivität: Task 4 liefert innerhalb der 2D-Gruppe eine Aufgabenkorrektheit von 100%, innerhalb der VR-Gruppe sind es 85,71% mit einer falsch gelösten Aufgabe. Der betroffene Proband ließ sich nicht alle Abhängigkeiten untereinander anzeigen, was über die entsprechende Aktivierung auf dem PDA möglich gewesen wäre. Stattdessen versuchte er, die Ports jeder einzelnen Insel zu selektieren. Dass dies der falsche Weg war, fiel ihm allerdings im Nachhinein selbst auf. So kommentierte er die Aufgabe mit "I did not find the right button! So I tried to connect all islands". Ein Proband der 2D-Gruppe bezeichnete die Aufgabe während der Durchführung als "gar nicht so einfach". Ein weiterer Proband kritisierte die fehlende Reset-Möglichkeit in der 2D-Anwendung ("Would be nice to have a reset feature, instead of deactivating previous selections").

Effizienz: Task 4 wurde von allen Anwendern in den beiden Gruppen schnell ausgeführt. Mit durchschnittlich 21,71 Sekunden war die 2D-Gruppe ein wenig schneller als die VR-Gruppe (33 Sekunden). Dies lässt sich auf die doch umständlichere Handhabung der VR-Anwendung zurückführen: Während die Probanden hier zunächst erst die nötige Handbewegung machen müssen, um sich das PDA anzeigen zu lassen, können die Anwender der 2D-Anwendung einfach im Drop Down-Menü der Informationsleiste die gewünsche Darstellung wählen. Die VR-Anwender müssen über das PDA zusätzlich zu dem richtigen Reiter mit den Einstellungen für die Dependencies gelangen und hier den gewünschten Haken setzen.

Zufriedenheit: Trotz der umständlicheren Handhabung der VR-Anwendung - insbesondere das Navigieren durch das PDA - bewertete die VR-Gruppe ihr System mit 18,43 Punkten besser als die 2D-Gruppe (16,71 Punkte). Ein VR-Proband bezeichnete die Aufgabe beispielsweise als "very fast and easy" und lobte vor allem das virtuelle Tablet ("Documentation was good, as the option to show all dependencies was shown directly on the PDA"). Generell wurde beobachtet, dass vor allem das dritte Item des ASQ (*"Overall, I am satisfied with the support information (online help, messages, documentation) when completing the task"*) deutlich bessere Bewertungen von

den Teilnehmern der VR-Gruppe erhielt (M = 5.57, SD = 1.81) als die 2D-Anwendung (M = 3.71, SD = 2.43), während die anderen zwei Items in beiden Gruppen sehr ähnliche Werte erzielten. Dies lässt sich dadurch begründen, dass das PDA vermutlich von den Probanden als Hilfe bzw. Dokumentation interpretiert wurde, während die Informationsleiste der 2D-Anwendung lediglich als Mittel zur Interaktion angesehen wurde, welches keinen weiteren Mehrwert für den Anwender liefert.

Design-Implikation: Das Ausführen dieser Aufgabe stellte für beide Gruppen keine bedeutenden Probleme dar. Alle Probanden lösten die Aufgabe zügig und mit fast vollständiger Korrektheit. Dennoch konnte beobachtet werden, dass die Anwendung über das virtuelle Tablet in Bezug auf die Zufriedenheit der Nutzer positiver angenommen wurde als die Auswahl über das Drop Down-Menü der 2D-Anwendung. Darüber hinaus wurde angemerkt, dass hier eine Reset-Möglichkeit hilfreich wäre, um die Darstellung aller Abhängigkeiten bzw. generell alle zuvor getätigten Einstellungen wieder zurückzusetzen. In der VR-Anwendung ist eine solche Funktion bereits implementiert. Außerdem hatten auch hier einige Probanden erneut Probleme, sich in der Informationsleiste zurecht zu finden, das heißt, der größte Anteil der benötigten Zeit war das Auffinden des Drop Down-Menüs. Zwar befindet sich dieses schon unter dem Punkt *Dependencies*, dennoch sollte auch hier die Darstellung insofern überarbeitet werden, dass der Anwender dabei unterstützt wird, die gewünschte Funktion intuitiv zu finden.

Ein Teilnehmer der VR-Gruppe konnte die Aufgabe nicht korrekt ausführen, da er den entsprechenden Tab - benannt als *Settings* - zum Einstellen der Abhängigkeiten nicht fand. Neben *Settings* existiert ein zweiter Tab namens *Inspect*, unter welchem die Informationen zu einer selektierten Komponente angezeigt werden, also beispielsweise der Name eines Bundles und die Anzahl der darin enthaltenen Packages. Hier könnte man die Registerkartenstruktur noch verbessern, da der Begriff *Settings* nicht direkt impliziert, dass sich hier die Einstellungen für Abhängigkeiten und Services befinden. Eine Möglichkeit wäre, eigene Tabs für beide Komponenten anzubieten und diese *Dependencies* und *Services* zu nennen.

Task 5

Lassen Sie sich alle Services bzw. Service Components anzeigen.

Effektivität: Task 5 ist die einzige von allen fünf Aufgaben, bei der sowohl die 2D- als auch die VR-Gruppe komplette Aufgabenkorrektheit mit jeweils 100% erreicht hat. Generell wurden hier auch keine weiteren Beobachtungen gemacht, da die meisten Probanden die Aufgabe zügig und eigenständig ausgeführt hatten. Ein Teilnehmer der 2D-Gruppe hatte zunächst noch ein Bundle selektiert, was dazu führte, dass nur die Services und Service Components zu diesem Bundle angezeigt wurden. Er bemerkte diesen Fehler aber und konnte die Aufgabe nach der Deselektierung dieses Bundles korrekt beenden. Dieser Teilnehmer war es auch, der folgenden Kritikpunkt anmerkte: "Existing selections are hard to see. A reset function or an information that something is selected would be nice."

Effizienz: Auch in Bezug auf die benötigte Zeit war dies die Aufgabe mit den besten Werten. Die VR-Gruppe war hier sogar geringfügig schneller mit 6,00 Sekunden im Vergleich zur 2D-Gruppe mit 6,86 Sekunden. Ein Teilnehmer bezeichnete die Aufgabe als "simple task that could be finished very quickly." Das - wenn auch nur geringfügig - schnellere Beenden der Aufgabe in der VR-Anwendung lässt sich darauf zurückführen, dass sich die Einstellungen für die Services und Service Components direkt bei den Einstellungen für die Abhängigkeiten befinden. Da das Anzeigen der Abhängigkeiten Ziel der vorhergehenden Aufgabe war, lassen sich hier Lerneffekte nicht ausschließen. In der 2D-Anwendung erfolgt das Anzeigen der Services und Service Components über eine eigene View und nicht - wie bei den Abhängigkeiten - über ein Drop Down-Menü.

Zufriedenheit: Die Nutzer der VR-Anwendung bewerteten diesen Task mit 18,86 Punkten besser als die 2D-Gruppe (16,71 Sekunden). Somit ist das insgesamt die Aufgabe mit den höchsten Werten, was mit den Ergebnissen aus Effektivität und Effizienz übereinstimmt. Unterschiede finden sich hier nur bei der Bewertung des dritten Items, das in der VR-Anwendung mit durchschnittlich 5,14 (SD = 2,27) und in der 2D-Anwendung nur mit 3,00 (SD = 2,20) Punkten bewertet wird. Zurückzuführen ist dies wieder - wie bereits in der vorherigen Aufgabe - auf die Verwendung des virtuellen Tablets, das die Nutzer der VR-Gruppe erneut als Unterstützung interpretieren.

Design-Implikation: Grundsätzlich lassen sich von den Ergebnissen dieser Aufgabe keine größeren Design-Implikationen ableiten. Wie schon bei Task 4 würde sich anbieten, für die Darstellung der Services in der VR-Anwendung einen separaten Tab auf dem PDA zu integrieren. Die Nutzer der 2D-Gruppe mussten in dieser Aufgabe die View wechseln, was sich aber für alle Probanden schnell und zielbringend ausführen ließ. Da hier die Bewertungen für Hilfsfunktionen verhältnismäßig niedrig ausfielen, könnte man durch eine ausführlichere Dokumentation oder Legende versuchen, den Nutzer zu unterstützen.

6.4 Zusammenfassung

Aus den Ergebnissen geht hervor, dass sich die Hypothesen in Bezug auf die Faktoren Effektivität, Effizienz und Zufriedenheit in zwei von drei Fällen als bestätigt erwiesen haben. In allen drei Fällen weist die 2D-Anwendung hinsichtlich der Usability bessere Ergebnisse auf, wovon jedoch der Unterschied bezüglich der Zufriedenheit als nicht signifikant angesehen werden muss. Auch innerhalb der Gruppen existieren Unterschiede hinsichtlich der durchgeführten Aufgaben. Bei beiden Anwendungen konnten so einige Implikationen für deren Weiterentwicklung gefunden werden.

Im Folgenden sollen nun noch die bisherigen Ergebnisse um die Gesamtbewertungen der beiden Anwendungen, die am Ende der Fragebögen erfasst wurden, ergänzt werden. Zur besseren Anschaulichkeit wurden die in den Bewertungen am häufigsten genannten Merkmale der Anwendungen in Tabelle 6.2 zusammengefasst.

Dabei bedeutet ein grüner Haken eine positive Erwähnung in der jeweiligen Kategorie, ein rotes Kreuz entspricht einer negativen Nennung. Kein Symbol bedeutet, dass keinerlei Auskunft über die jeweilige Kategorie gegeben wurde.

	2D	VR
Überblick über Softwareprojekt	✓	✗✗
Übersichtliche Darstellung	✓✗✗	✓
Darstellung der Abhängigkeiten	✓	✓
Darstellung der wichtigsten Komponenten	✓	✓
Informationen über Komponenten	✗	✗✗✗
Leicht verständliche Darstellung	✗	✗✗
Schnelle Aufgabenerfüllung	✗	✗✗
Macht Spaß		✓
Einfache Handhabung		✗
Intuitive Bedienung		✗

Tabelle 6.2 — Die am häufigsten genannten Aspekte aus der Gesamtbewertung.

Das Ziel, einen ersten Überblick über ein Softwareprojekt zu geben, wurde von einem Probanden der 2D-Anwendung positiv erwähnt: "Die Anwendung halte ich für geeignet um einen groben Überblick zu erhalten [...]". Bei der VR-Anwendung hingegen wurde dieser Punkt zweimal infrage gestellt. So erwähnte ein Teilnehmer: "Die Visualisierung scheint wenig zu nutzen, um die Software kennenzulernen." Ein anderer gab an, die Visualisierung helfe zwar dabei, das Konzept von OSGi zu verstehen, aber nicht das Projekt im Ganzen.

Hinsichtlich der Übersichtlichkeit der Darstellung wurden sowohl die 2D als auch die VR-Anwendung in jeweils einem Fall positiv erwähnt. Zwei Teilnehmer der 2D-Gruppe revidierten das jedoch: "Die Darstellung der Bundles und Services ist bei der hohen Anzahl an Nodes schnell sehr komplex und wird unübersichtlich." Die zweite negative Nennung bezeichnete die Aufteilung als unübersichtlich.

Ein weiterer Punkt, der in beiden Anwendungen jeweils einmal erwähnt wurde, ist die Darstellung der Abhängigkeiten zwischen den Bundles. Ein Proband der VR-Anwendung hob dabei die Abbildung durch Pfeile positiv hervor ("Gut sind auch die Input/Output-Pfeile, die darstellen wie die Abhängigkeiten der verschiedenen Bundles sind"). Eine Teilnehmerin in der 2D-Gruppe gab an, ihr gefalle die Darstellung der Abhängigkeiten.

Die VR-Anwendung scheint bezüglich der Informationsgewinnung über die einzelnen Komponenten noch einige Defizite aufzuweisen. Insgesamt dreimal wurde dieser Punkt negativ erwähnt. Dabei wurde stets betont, dass wenige Informationen zu den Bundles, Packages und Klassen angezeigt werden. Ein Teilnehmer gab ein spezifisches Beispiel dazu an ("Some extra information would be nice, e.g. what are the functions of the different bundles, packages, classes? How are they interacting with each other?"). Die 2D-Anwendung weist in diesem Bereich laut einem Probanden noch Verbesserungsbedarf auf: "Die Anwendung halte ich für geeignet einen groben Überblick zu erhalten [...], jedoch nicht um detaillierte Daten über einzelne Objekte zu erfahren."

In Bezug darauf, ob die Anwendungen leicht verständlich dargestellt sind, müssen insgesamt drei Negativnennungen erwähnt werden. In der Gruppe der 2D-Visualisierung bemängelt ein Teilnehmer, die Anwendung sei ohne Einführung schwer zu verwenden. In der VR-Anwendung wurden Verständnisprobleme genannt, die in diese Kategorie gezählt werden. Ein Teilnehmer hatte beispielsweise Probleme, die Abstände zwischen den Inseln einzuschätzen: "It was unclear whether or not islands that are close together are related; is there any relationship between close islands or is it just random?" Dieser Punkt wurde von einem weiteren Probanden der VR-Gruppe genannt: "Es wäre hilfreich wenn es eine "Help"-Schaltfläche geben würde, sodass man kurz Navigationshilfe bekommt wenn man nicht weiter weiß. Es wäre auch nett zu wissen, warum gewisse Bundles so einen großen Abstand zueinander haben und andere nicht."

Auch die Aufgabenerfüllung wurde erwähnt. So werden insgesamt drei Negativnennungen in beiden Anwendungen gezählt. Ein Proband der 2D-Visualisierung gab an, dass "komplexere Aufgabenstellungen unnötig viel Zeit [benötigten] aufgrund der unübersichtlichen Darstellung". In der VR-Anwendung notierte ein Teilnehmer: "A search function would be really useful to complete tasks like "locate package x". Hier wurde zwar nicht explizit erwähnt, dass das Lösen einer Aufgabe unnötig viel Zeit in Anspruch nahm; die Aussage impliziert aber, dass durch eine solche Suchhilfe die Aufgaben weitaus einfacher und schneller zu erledigen wären. Ein anderer statuierte: "Einige Inseln sind sehr weit entfernt, weshalb man etwas Zeit benötigt, um zu ihnen zu gelangen."

Spaß während der Anwendung ist ein weiterer Faktor, die die Zufriedenheit der Nutzer beeinflusst. In der Gesamtbewertung wurde dieser Punkt allerdings nur von einem Anwender der VR-Gruppe genannt.

Bezüglich der Handhabung wurde eine Negativnennung in der VR-Gruppe gezählt. Dabei bezeichnete der Teilnehmer das Zoomen und Verschieben innerhalb der Anwendung als "flimsy". Auch dass die Anwendung nicht intuitiv anzuwenden ist, wurde einmal erwähnt: "[...] not intuitive enough for the first use (which was use case)."

In Summe werden in beiden Anwendungen jeweils viermal Merkmale der oben genannten Kategorien positiv hervorgeheben; Negativnennungen kommen in der 2D-Anwendung fünfmal vor, in der VR-Anwendung sind es deutlich mehr: Insgesamt elfmal finden sich hier Kritikpunkte. Die Ergebnisse der Gesamtbewertung sind somit deckungsgleich mit den Ergebnissen, die aus den Messungen der quantitativen Daten sowie der Fragebögen hervorgeht.

7 Diskussion und Ausblick

7.1 Gewonnene Erkenntnisse

Die im Rahmen der vorliegenden Arbeit durchgeführte Usability-Studie über zwei Softwarevisualisierungen in 2D und VR brachte viele neue Erkenntnisse hervor. Zum einen konnten definitiv Schwachstellen der Systeme aufgedeckt werden, was das übergeordnete Ziel dieser Arbeit war. Sie weist allerdings auch einige Limitationen auf, die im Folgenden vorgestellt werden und als Implikationen für zukünftige Forschungsarbeiten dienen.

Zunächst sollte erwähnt werden, dass das Studiendesign einer Vergleichsstudie zwischen zwei Systemen, die in ihren Funktionalitäten teilweise sehr verschieden sind, gewisse Defizite aufweist. Da die Probanden der beiden Gruppen innerhalb ihrer Anwendung diesselben Tasks durchführen sollten, mussten Aufgaben gewählt werden, die in beiden Anwendungen gleichermaßen zu erledigen waren. Dies führte jedoch dazu, dass einige Merkmale, die in der jeweils anderen Anwendung nicht implementiert waren, dementsprechend auch nicht getestet werden konnten. Beispielsweise weist die 2D-Anwendung die Möglichkeit auf, ein selektiertes Bundle zu einer Treemap zu expandieren. Dadurch kann sich der Anwender einen ersten Überblick über die innere Struktur eines Bundles verschaffen. Eine vergleichsweise Darstellung existiert in der VR-Anwendung jedoch nicht; hier wird die innere Struktur lediglich durch die Inselmetapher mit ihren Regionen und Gebäuden dargestellt. Dementsprechend war es nicht möglich, die Verwendung der Treemap in die Aufgabenstellung miteinzubeziehen.

Des Weiteren sollte die unterschiedliche Bedienbarkeit der Anwendungen berücksichtigt werden. Während die Probanden der 2D-Gruppe ihre Aufgaben in einem größtenteils vertrauten Arbeitsumfeld durchführten - bestehend aus einem Schreibtisch mit Laptop und angeschlossener Maus - mussten sich die Probanden der VR-Gruppe einer überwiegend komplett neuen Umgebung anpassen, da die meisten Teilnehmer bisher keine bis wenig Erfahrungen mit Virtual Reality gesammelt hatten. Auch typische Limitationen, die Virtual Reality mit sich bringt - beispielsweise physische Einschränkungen wie das mit dem Computer verbundene Kabel der VR-Brille - wurden bei dem vorliegenden Studiendesign nicht tiefergehend berücksichtigt. Auch andere Risiken wie kognitive Überlastung oder Motion Sickness wurden nicht geprüft. Zwar konnte beobachtet werden, dass einige Probanden durch die ungewohnte Situation zunächst verunsichert wirkten - durch die ausführliche Einführung und Erklärung der Anwendung sollte dem entgegengewirkt werden. Dennoch muss angenommen werden, dass die durch die VR-Anwendung hervorgerufene Immersion Auswirkungen auf die Studienergebnisse hat.

Eine weitere Limitation der innerhalb dieser Arbeit durchgeführten Studie ist die Prüfung der Verständnisgewinnung über ein bestehendes Softwareprojekt. Zwar handelt es sich um eine Usability-Studie, die in erster Linie nur auf die Evaluation der Benutzerfreundlichkeit der Bedienoberfläche abzielt. In diesem Bereich konnten durch die Studie wertvolle Erkenntnisse gewonnen werden. Da Softwarevisualisierungen aber übergeordnet die Verständnisgewinnung sowie das Vermitteln eines ersten Überblicks über ein bestehendes Softwareprojekt zum Ziel haben, sollte dies auch durch ein entsprechendes Studiendesign abgefragt werden.

Die Reihenfolge und Formulierung der im Rahmen der Studie durchgeführten Tasks stellt einen zusätzlichen Einflussfaktor auf die Ergebnisse dar. Die beiden letzten Aufgaben ähneln sich in ihrer Durchführung recht stark, was möglicherweise zu Lerneffekten bei den Teilnehmern geführt hat. Deshalb sind gerade die Auswertungsergebnisse von Task 5 kritisch zu betrachten. Dies konnte vor allem innerhalb der VR-Gruppe beobachtet werden, da die Durchführung der Aufgabe exakt demselben Vorgehen entspricht wie bei Task 4. Während der Vorbereitungsphase wurde dies zwar bereits durch das Pretesting festgestellt. Zunächst war die Überlegung, die beiden Aufgaben jeweils an den Anfang und das Ende des Taskflows zu stellen. Da die Aufgabenstellung aber einer induktiven Systematik folgte, sollten die drei Hauptkomponenten Bundles, Packages und Klassen sowie die Übersichtdarstellungen der Abhängkeiten und Services in der Reihenfolge als zusammengehörig erkennbar sein.

Die Formulierung der Aufgabenstellung stellte gerade für die Probanden der VR-Anwendung teilweise ein Problem dar, da ihnen - mit Ausnahme von Task 3, bei dem ein spezifisches Bundle gefunden werden sollte - die Aufgabenstellungen vorgelesen wurde. Vor allem die zweite Aufgabe wurde aufgrund der ausgedehnteren Syntax häufig nicht richtig oder erst auf Nachfrage verstanden.

Als weitere Limitation der Studie ist die Repräsentativität der ungleichmäßigen Stichprobe zu nennen. Wie bereits unter Kapitel 6 erwähnt, sind unter den insgesamt 14 Teilnehmern 64,3% und somit die Mehrheit der Probanden männlich. Mit der durch weitere 64,3% erfassten Altersgruppe zwischen 25 und 34 Jahren wird eine recht junge Stichprobe repräsentiert, die damit der Generation der Digital Natives angehört. Eine größere Anzahl an Probanden höheren Alters könnte eine Veränderung der Ergebnisse vor allem in Bezug auf Effizienz und Zufriedenheit hervorrufen.

7.2 Zukünftige Entwicklung

Softwarevisualisierungen sind eine gewinnbringende Möglichkeit, die Komplexität von Source Code zu verringern und ihn dadurch leichter verständlich zu machen. Eine wichtigte Grundvoraussetzung dafür ist jedoch auch eine gute Usability, die zu einem positiven Nutzungserlebnis beiträgt und deshalb ein wichtiger Faktor für das Erreichen der Ziele von Softwarevisualisierungen ist.

Die vorliegende Arbeit liefert eine erste Evaluierung der Usability über zwei auf OSGi basierenden Softwarevisualisierungen in 2D und VR, was einen neuen Ansatz in diesem Bereich darstellt. Der Fokus lag dabei auf dem Erreichen möglichst hoher Werte im Bereich Effektivität, Effizienz und Zufriedenheit. In allen drei Punkten konnte die 2D-Anwendung bessere Werte erzielen als die VR-Anwendung. Die Gründe dafür sind vielfältig:

Zunächst ist die Nutzung von VR-Anwendungen für die meisten Personen nach wie vor eine neue und ungewohnte Erfahrung. Gerade im Alltag werden solche Systeme bisher kaum genutzt. Trotz großer technischer Fortschritte in den letzten Jahren zählen die physischen Beeinträchtigungen während der Nutzung nach wie vor zu den Hauptproblemen. So führt das Aufsetzen des meist klobigen und schweren Headsets häufig zu einem unbequemen Tragefühl und kann darüber hinaus Konsequenzen wie Nacken- oder Kopfschmerzen nach sich ziehen. Die meisten VR-Systeme sind außerdem nach wie vor über Kabel mit den nötigen Rechnern verbunden, was zu einer Einschränkung der Bewegungsfreiheit führt. Damit einhergehend verspüren viele Nutzer Hemmungen bei der Nutzung eines unbekannten Systems, was dazu führt, dass sie sich kognitiv nicht komplett auf die Anwendung einlassen können. Auch die Darstellung der virtuellen Realität kann Probleme bereiten. So bewirkt beispielsweise die sogenannte *Motion Sickness* Schwindel und Übelkeit beim Nutzer, ausgelöst durch zeitliche Diskrepanzen zwischen der visuellen Darstellung und der Kopfbewegung des Anwenders. Um dem entgegenzuwirken, müssen die VR-Systeme zukünftig höhere Pixeldichten und Bildfrequenzen aufweisen.

Bei der Anwendung jeglicher Art von Computersystemen muss jedoch stets der jeweilige Nutzungskontext berücksichtigt werden. Dies ist gerade bei der Usabilityevaluierung ein maßgeblicher Aspekt, da sich die Ziele und Zielgruppen solcher Systeme häufig stark unterscheiden. Die Evaluation von Softwarevisualisierungen, wie sie in dieser Arbeit vorliegen, erfordert natürlich völlig andere Aspekte der Betrachtung als eine VR-Anwendung, die beispielsweise für die Unterhaltungsbranche entwickelt wurde. Für zukünftige Forschungsarbeiten in diesem Bereich sollte außerdem - vor allem bei der Evaluierung von VR-Anwendungen - der Einsatz von technischen Messinstrumenten miteinbezogen werden. So können beispielswiese Eyetracking-Verfahren viel mehr Aufschluss darüber geben, wie eine Person die virtuelle Realität wahrnimmt. Durch gezieltes Nachverfolgen der Augenbewegungen können sich so neue und hilfreiche Ansätze für die Entwicklung solcher Systeme ergeben.

Darüber hinaus sollte der Einsatz von VR-Systemen für den vorliegenden Use Case hinterfragt werden. Zwar kann es für einen Softwareentwickler durchaus hilfreich sein, sich durch die Exploration einer solchen virtuellen Welt einen ersten Überblick über die Komplexität und den Umfang eines Softwareprojekts zu verschaffen. Langfristig wird er aber mit dem reinen Source Code mehr anfangen können. Für Anwender aus anderen Bereichen, wie beispielsweise Projektleiter, eignet sich der Ansatz für einen ersten Einstieg recht gut. Allerdings sollte hier verstärkt über den Einsatz von Augmented Reality nachgedacht werden, da sich diese Technologie gerade für kollaboratives Arbeiten eignet. Dies könnte einen erheblichen Mehrwert mit sich bringen: So könnten sich Softwareentwickler sowie fachfremde Personen, die aber in dasselbe Projekt involviert sind, viel leichter austauschen, was eine Verständnisgewinnung maßgeblich unterstützen würde.

Aufgrund der in dieser Arbeit gewonnenen Erkenntnisse lässt sich festhalten, dass die evaluierte VR-Anwendung in ihrer jetzigen Form noch diverse Schwachstellen aufweist. Auch bei der browserbasierten Visualisierung in 2D besteht Verbesserungsbedarf, auch wenn die Ergebnisse zeigen, dass sich die Probanden hier effektiver und effizienter zurecht finden konnten. Deshalb eignet sich für zukünftige Forschungsarbeiten die Darstellung von Softwarevisualisierungen in Augmented Reality, die als eine Kombination aus beiden Ansätzen gesehen werden kann: Das gewohnte Umfeld des Anwenders um virtuelle Elemente ergänzen sowie zusätzliche Unterstützung durch weitere Personen ermöglichen. Das Hauptziel von Softwarevisualisierungen kann dadurch nachhaltiger erreicht werden.

Literaturverzeichnis

[1] Bailey, Brian P.: *Psychology of HCI*, 2004.

[2] Balzer, Michael und Oliver Deussen: *Level-of-Detail Visualization of Clustered Graph Layouts*. Asia-Pacific Symposium on Visualization, Februar 2007.

[3] Bangor, Aaron, Philip Kortum und James Miller: *Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale*. Journal of Usability Studies, 4:114–123, Mai 2009.

[4] Barth, Michael: *Prinzipien der Modularisierung*. Universität Ulm, 2012.

[5] Bohnet, Johannes und Jürgen Döllner: *Grundlagen der Softwarevisualisierung*. In: *Konzepte der Softwarevisualisierung für komplexe, objektorientierte Softwaresysteme*, Band 6, Kapitel 1, Seiten 3–7. Hasso-Plattner-Institut für Softwaresystemtechnik Potsdam, 2005.

[6] Brooke, John: *SUS: A quick and dirty Usability Scale*. In: Jordan, Patrick W., Bruce Thomas, Ian Lyall McClelland und Bernard Weerdmeester (Herausgeber): *Usability Evaluation in Industry*. CRC Press, Juni 1996.

[7] Brosius, Felix: *SPSS für Dummies*. WILEYVCH Verlag GmbH Co. KGaA, Weinheim, 1. Auflage, 2017.

[8] Card, Stuart K., Thomas P. Moran und Allen Newell: *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, NJ [u.a.], 1983.

[9] Caserta, Pierre und Olivier Zendra: *Visualization of the Static Aspects of Software: A Survey*. IEEE Transactions on Visualization and Computer Graphics, 17(7):913–933, July 2011.

[10] Churcher, Neville, Warwick Irwin und Ron Kriz: *Visualizing Class Cohesion with Virtual Worlds*. In: *Proceedings of the Asia-Pacific Symposium on Information Visualisation - Volume 24*, APVis '03, Seiten 89–97. Australian Computer Society, Inc., 2003.

[11] Diehl, Stephan: *Software Visualization. Visualizing the Structure, Behaviour and Evolution of Software*. Springer-Verlag Berlin Heidelberg, 2007.

[12] DIN EN ISO 6385: *Grundsätze der Ergonomie für die Gestaltung von Arbeitssystemen (ISO 6385:2016)*, Dezember 2016.

[13] DIN ISO 9241-11: *Ergonomie der Mensch-System-Interaktion - Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte (ISO/DIS 9241-11.2:2016)*, Dezember 2016.

[14] Ducasse, Stéphane und Michele Lanza: *The Class Blueprint: Visually Supporting the Understanding of Classes.* IEEE Transactions on Software Engineering, 31(1):75–90, Januar 2005.

[15] Eick, Stephen, Joseph Steffen und Eric Sumner: *Seesoft - A Tool for Visualizing Line Oriented Software Statistics.* IEEE Transactions on Software Engineering, 18(11):957–968, November 1992.

[16] Eid, Michael, Mario Gollwitzer und Manfred Schmitt: *Statistik und Forschungsmethoden*, Band 4 der Reihe *10.* Beltz, Weinheim, 5 Auflage, 2017.

[17] Field, Andy: *Discovering Statistics Using IBM SPSS Statistics.* SAGE Publications, Los Angeles [u.a.], 4. Auflage, 2013.

[18] Gračanin, Denis, Kresimir Matkovic und Mohamed Eltoweissy: *Software Visualization.* Innovations in Systems and Software Engineering, 1(2):221–230, September 2005.

[19] Hagedorn, Benjamin: *Landschafts- und Stadtmetaphern zur Softwarevisualisierung.* In: *Konzepte der Softwarevisualisierung für komplexe, objektorientierte Softwaresysteme*, Band 6, Kapitel 7, Seiten 61–72. Hasso-Plattner-Institut für Softwaresystemtechnik Potsdam, 2005.

[20] Holten, Danny: *Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data.* IEEE Transactions on Visualization and Computer Graphics, 12, September 2006.

[21] Knight, Claire und Malcom Munro: *Comprehension with[in] Virtual Environment Visualisations.* Juli 1999.

[22] Lavery, Darryn, Gilbert Cockton und Malcolm P. Atkinson: *Comparison of evaluation methods using structured usability problem reports.* Behaviour & Information Technology, 16(4-5):246–266, 1997.

[23] Lewis, James R.: *Psychometric Evaluation of an After-scenario Questionnaire for Computer Usability Studies: The ASQ.* SIGCHI Bull., 23(1):78–81, Januar 1991.

[24] Maletic, Jonathan I., Jason Leigh und Andrian Marcus: *Visualizing Software in an Immersive Virtual Reality Environment.* In: *Proceedings of ICSE'01 Workshop on Software Visualization*, Seiten 12–13. Society Press, 2001.

[25] Maletic, Jonathan I., Andrian Marcus und Michael L. Collard: *A Task Oriented View Of Software Visualization.* In: *Proceedings of IEEE Workshop of Visualizing Software for Understanding and Analysis (VISSOFT 2002)*, Seiten 32–40, Paris, Juni 2002. IEEE Working Conference on Software Visualization, IEEE.

[26] Marcus, Andrian, Denise Comorski und Andrey Sergeyev: *Supporting the Evolution of a Software Visualization Tool Through Usability Studies.* In: *Proceedings of the 13th International Workshop on Program Comprehension (IWPC 2005)*, Seiten 307–316. IEEE, 2005.

[27] Marcus, Andrian, Louis Feng und Jonathan I. Maletic: *3D Representations for Software Visualization.* In: *Proceedings of the ACM Symposium on Software Visualization (SoftVis)*, San Diego, 2003. ACM.

[28] Marquardt, Tobias: *Extraktion und Visualisierung von Beziehungen und Abhängigkeiten zwischen Komponenten großer Softwareprojekte.* Diplomarbeit, Technische Universität Dortmund, April 2016.

[29] McAffer, Jeff, Paul VanderLei und Simon Archer: *OSGi and Equinox: Creating Highly Modular Java Systems.* Addison-Wesley Professional, 1. Auflage, 2010.

[30] Moser, Christian: *User Experience Design: Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern.* Springer-Verlag, Berlin Heidelberg, 2012.

[31] Murray, Scott: *Interactive Data Visualization for the Web.* O'Reilly Media, Sebastopol, 2013.

[32] Nielsen, Jakob: *Why You Only Need to Test with 5 Users.* Nielsen Norman Group, März 2000. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/, aufgerufen am 11.03.2019.

[33] Nielsen, Jakob und Ralf Molich: *Heuristic Evaluation of User Interfaces.* In: Chew, Jane Carrasco und John Whiteside (Herausgeber): *CHI '90 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM New York, April 1990.

[34] Norman, Don: *The Design of Everyday Things.* Basic Books, New York, Revised Auflage, 2013.

[35] OSGi Alliance: *OSGi Core Release 7.* OSGi Alliance, April 2018. URL: https://osgi.org/specification/osgi.core/7.0.0/index.html, aufgerufen am 11.10.2018.

[36] Panas, Thomas, Rebecca Berrigan und John Grundy: *A 3D metaphor for software production visualization.* In: *Proceedings of the Seventh International Conference on Information Visualization (IV'03)*, Seiten 314 – 319. IV 2003 - Seventh International Conference on Information Visualization, August 2003.

[37] Polson, Peter G., Clayton Lewis, John Rieman und Cathleen Wharton: *Cognitive walkthroughs: a method for theory-based evaluation of user interfaces.* International Journal of Man-Machine Studies, 36:741–773, Mai 1992.

[38] Preim, Bernhard und Raimund Dachselt: *Interaktive Systeme. Band 1: Grundlagen, Graphical User Interfaces, Informationsvisualisierung.* Springer-Verlag, Berlin Heidelberg, 2. Auflage, 1999.

[39] Pressman, Roger S.: *Software Engineering: A Practitioner's Approach.* McGraw Hill, Boston, 7. Auflage, 2010.

[40] Prothero, Jerrold, M. H. Draper, Thomas Furness, Don E. Parker und M. J. Wells: *The use of an independent visual background to reduce simulator side-effects.* Aviation, Space and Environmental Medicine, 70:277–283, April 1999.

[41] Rasch, Björn, Malte Friese, Wilhelm Hofmann und Ewald Naumann: *Quantitative Methoden 2. Einführung in die Statistik für Psychologen und Sozialwissenschaftler.* Springer-Verlag, Heidelberg, 2. Auflage, 2006.

[42] Reussner, Ralf: *Handbuch der Software-Architektur.* dpunkt.verlag, Heidelberg, 2. Auflage, 2009.

[43] Rohrer, Christian: *When to Use Which User-Experience Research Methods.* Nielsen Norman Group, Oktober 2014. URL: https://www.nngroup.com/articles/which-ux-research-methods/, aufgerufen am 17.01.2019.

[44] Rössler, Patrick: *Inhaltsanalyse.* UTB GmbH, 2010.

[45] Santos, Cristina Russo Dos, Pascal Gros, Pierre Abel, Didier Loisel, N. Trichaud und J. P. Paris: *Metaphor-Aware 3D Navigation.* In: *Proceedings of the IEEE Symposium on Information Visualization 2000*, Seiten 155–165, Salt Lake City, Oktober 2000.

[46] Sarodnick, Florian und Henning Brau: *Methoden der Usability Evaluation. Wissenschaftliche Grundlagen und praktische Anwendung.* Verlag Hans Huber, Bern, 2. Auflage, 2011.

[47] Sauro, Jeff und James R. Lewis: *Quantifying the User Experience : Practical Statistics for User Research.* Morgan Kaufmann, Amsterdam, March 2012.

[48] Seider, Doreen: *Open Source Framework RCE: Integration, Automation, Collaboration.* 4th Symposium on Collaboration in Aircraft Design, November 2014.

[49] Seider, Doreen, Tobias Marquardt, Andreas Schreiber und Marlene Brüggemann: *Visualizing Modules and Dependencies of OSGi-based Applications.* In: *Software Visualization (VISSOFT)*, Seiten 96–100. IEEE Working Conference on Software Visualization, IEEE, 2016.

[50] Shackel, Brian: *Ergonomics for a Computer.* Design, 120:36–39, 1959.

[51] Sommerville, Ian: *Software Engineering.* Pearson Addison Wesley, 7. Auflage, 2004.

[52] Stapelkamp, Torsten: *Interaction- und Interfacedesign: Web-, Game-, Produkt- und Systemdesign; Usability und Interface als Corporate Identity.* Springer, Berlin [u.a.], 2010.

[53] Telea, Alexandru: *Data Visualization: Principles and Practice.* CRC Press, 2. Auflage, September 2014.

[54] Telea, Alexandru und Lucian Voinea: *A Framework for Interactive Visualization of Component-Based Software.* In: *EPRINTS-BOOK-TITLE University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science*, Seiten 567–574. Proc. 30th EUROMICRO Conf. (EUROMICRO '04), 2004.

[55] Tullis, Tom und Bill Albert: *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics.* Morgan Kaufmann, Bern, 2. Auflage, 2013.

[56] Wandmacher, Jens: *Software-Ergonomie.* De Gruyter, 1. Auflage, 1993.

[57] Ward, Matthew, Georges Grinstein und Daniel Keim: *Interactive Data Visualization: Foundations, Techniques, and Applications.* CRC Press, 2. Auflage, June 2015.

[58] Ware, Colin und Glenn Franck: *Viewing a Graph in a Virtual Reality Display is Three Times as Good as a 2D Diagram.* In: *Visual Languages*, Seiten 182–183, Oktober 1994.

[59] Wütherich, Gerd, Nils Hartmann, Bernd Kolb und Matthias Lübken: *Die OSGI Service Platform. Eine Einführung mit Eclipse Equinox.* dpunkt.verlag, 1. Auflage, 2008.

[60] Yang, Hong Yul, Hamish Graham und Rebecca Berrigan: *Software Metrics and Visualisation.* University of Auckland, Department of Electrical and Electronic Engineering (Software Engineering), 2003.

[61] Young, Peter und Malcom Munro: *Visualizing Software in Virtual Reality.* In: *6th International Workshop on Program Comprehension (IWPC '98)*, Januar 1998.

Abbildungsverzeichnis

Tabellenverzeichnis

A Einführungstext zur Studie

STUDIE:
USABILITY VON SOFTWAREVISUALISIERUNGEN
IN 2D & VIRTUAL REALITY

Lieber Teilnehmer,

vielen Dank für Ihr Interesse und Ihre Mithilfe an der Usability-Studie für meine Masterarbeit.

Im Folgenden liegt Ihnen ein Use Case vor, den Sie sich bitte aufmerksam durchlesen. Anschließend bekommen Sie einige Minuten Zeit, um sich mit der grundsätzlichen Benutzung der Anwendung vertraut zu machen.

Die eigentliche Studie besteht aus 5 Aufgaben (*Tasks*), die Sie der Reihe nach durchführen werden. Nach jedem durchgeführen Task werden Sie die Aufgabe mit einem kurzen Fragebogen bewerten.

Am Ende erfolgt noch das Ausfüllen eines Fragebogens, der auf die Bewertung des Gesamtsystems abzielt.

Viel Spaß!

Sie haben vor wenigen Tagen Ihre neue Stelle als Softwareentwickler eines Ihnen bisher noch recht unbekannten Projekts angetreten. Sie wissen zwar grob, dass es sich bei RCE um eine Integrationsumgebung für Wissenschaftler und Ingenieure zum Analysieren, Entwerfen und Simulieren komplexer Systeme handelt; Einblick in die Softwarearchitektur von RCE hatten Sie aber bisher noch nicht. Den möchten Sie sich jetzt aber möglichst schnell gewinnen, umfasst ihre neue Stelle doch unter anderem die Weiterentwicklung und Wartung der Software.

Einige Kollegen in Ihrer Abteilung forschen im Bereich der Softwarevisualisierung und möchten so Softwarecode greifbarer machen. Zufälligerweise wurde auch RCE, welches auf dem OSGi-Framework basiert, visualisiert.

Als erfahrener Softwarenentwickler kennen Sie zwar den auf Java basierenden OSGi-Framework, der es ermöglicht, Softwarecode in die drei Komponenten Bundles, Packages und Classes zu unterteilen. Die Darstellung von Software als 2D- / VR-Visualisierung ist Ihnen jedoch gänzlich neu.

Das Softwareprojekt, bei dem Sie ab sofort mitarbeiten werden, ist sehr umfangreich und besteht aus vielen verschiedenen Komponenten. Mithilfe der von Ihren Kollegen entwickelten Visualisierung möchten Sie sich nun einen ersten Überblick verschaffen.

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

Task 1:

Selektieren Sie das Bundle mit der größten Anzahl an Packages.
Benennen Sie die Anzahl der Packages.

Task 2:

Selektieren Sie ein beliebiges Bundle.
Nennen Sie den Namen des am weitesten entfernten Bundles, zu dem eine Verbindung besteht, und die Anzahl der darin enthaltenen Klassen.

Task 3:

Selektieren Sie das Bundle *RCE Core Utils Scripting.*
Benennen Sie die Anzahl der Imports und Exports.

Task 4:

Lassen Sie sich die Abhängigkeiten aller Bundles untereinander anzeigen.

Task 5:

Lassen Sie sich alle Services bzw. Service Components anzeigen.

B SPSS-Ausgaben

Häufigkeitstabellen zur Soziodemographie

Statistiken

		Gender	Age	Education	Java expertise
N	Gültig	14	14	14	14
	Fehlend	0	0	0	0

Häufigkeitstabelle

Gender

		Häufigkeit	Prozent	Gültige Prozente	Kumulierte Prozente
Gültig	female	5	35,7	35,7	35,7
	male	9	64,3	64,3	100,0
	Gesamt	14	100,0	100,0	

Age

		Häufigkeit	Prozent	Gültige Prozente	Kumulierte Prozente
Gültig	18-24	4	28,6	28,6	28,6
	25-34	9	64,3	64,3	92,9
	35-44	1	7,1	7,1	100,0
	Gesamt	14	100,0	100,0	

Education

		Häufigkeit	Prozent	Gültige Prozente	Kumulierte Prozente
Gültig	High school graduate, diploma or the equivalent	7	50,0	50,0	50,0
	Bachelor`s degree	3	21,4	21,4	71,4
	Master`s degree	2	14,3	14,3	85,7
	Doctorate degree	2	14,3	14,3	100,0
	Gesamt	14	100,0	100,0	

Java expertise

		Häufigkeit	Prozent	Gültige Prozente	Kumulierte Prozente
Gültig	Basic	7	50,0	50,0	50,0
	Advanced	7	50,0	50,0	100,0
	Gesamt	14	100,0	100,0	

T-Test für Task Success

T-Test

Gruppenstatistiken

	Gruppe	N	Mittelwert	Std.-Abweichung	Standardfehler des Mittelwertes
TaskSuccess_ges	2D	7	97,14	7,559	2,857
	VR	7	82,86	7,559	2,857

Test bei unabhängigen Stichproben

		Levene-Test der Varianzgleichheit		T-Test für die Mittelwertgleichheit				
		F	Signifikanz	T	df	Sig. (2-seitig)	Mittlere Differenz	Standardfehler der Differenz
TaskSuccess_ges	Varianzen sind gleich	,000	1,000	3,536	12	,004	14,286	4,041
	Varianzen sind nicht gleich			3,536	12,000	,004	14,286	4,041

Test bei unabhängigen Stichproben

		T-Test für die Mittelwertgleichheit	
		95% Konfidenzintervall der Differenz	
		Untere	Obere
TaskSuccess_ges	Varianzen sind gleich	5,482	23,089
	Varianzen sind nicht gleich	5,482	23,089

T-Test für Time

T-Test

Gruppenstatistiken

	Gruppe	N	Mittelwert	Std.-Abweichung	Standardfehler des Mittelwertes
Time_ges	2D	7	212,43	77,123	29,150
	VR	7	514,00	175,895	66,482

Test bei unabhängigen Stichproben

		Levene-Test der Varianzgleichheit		T-Test für die Mittelwertgleichheit				
		F	Signifikanz	T	df	Sig. (2-seitig)	Mittlere Differenz	Standardfehler der Differenz
Time_ges	Varianzen sind gleich	5,727	,034	-4,154	12	,001	-301,571	72,592
	Varianzen sind nicht gleich			-4,154	8,225	,003	-301,571	72,592

Test bei unabhängigen Stichproben

		T-Test für die Mittelwertgleichheit	
		95% Konfidenzintervall der Differenz	
		Untere	Obere
Time_ges	Varianzen sind gleich	-459,735	-143,408
	Varianzen sind nicht gleich	-468,175	-134,967

T-Test für ASQ

T-Test

Gruppenstatistiken

	Gruppe	N	Mittelwert	Std.-Abweichung	Standardfehler des Mittelwertes
Satisfaction_ASQ	2D	7	77,14	11,824	4,469
	VR	7	77,86	14,100	5,329

Test bei unabhängigen Stichproben

		Levene-Test der Varianzgleichheit		T-Test für die Mittelwertgleichheit				
		F	Signifikanz	T	df	Sig. (2-seitig)	Mittlere Differenz	Standardfehler der Differenz
Satisfaction_ASQ	Varianzen sind gleich	,198	,664	-,103	12	,920	-,714	6,955
	Varianzen sind nicht gleich			-,103	11,646	,920	-,714	6,955

Test bei unabhängigen Stichproben

		T-Test für die Mittelwertgleichheit	
		95% Konfidenzintervall der Differenz	
		Untere	Obere
Satisfaction_ASQ	Varianzen sind gleich	-15,868	14,440
	Varianzen sind nicht gleich	-15,919	14,491

T-Test für SUS

T-Test

Gruppenstatistiken

	Gruppe	N	Mittelwert	Std.-Abweichung	Standardfehler des Mittelwertes
Satisfaction_SUS	2D	7	54,286	9,9702	3,7684
	VR	7	49,643	8,2195	3,1067

Test bei unabhängigen Stichproben

		Levene-Test der Varianzgleichheit		T-Test für die Mittelwertgleichheit				
		F	Signifikanz	T	df	Sig. (2-seitig)	Mittlere Differenz	Standardfehler der Differenz
Satisfaction_SUS	Varianzen sind gleich	,142	,713	,951	12	,361	4,6429	4,8839
	Varianzen sind nicht gleich			,951	11,579	,361	4,6429	4,8839

Test bei unabhängigen Stichproben

		T-Test für die Mittelwertgleichheit	
		95% Konfidenzintervall der Differenz	
		Untere	Obere
Satisfaction_SUS	Varianzen sind gleich	-5,9981	15,2839
	Varianzen sind nicht gleich	-6,0412	15,3270

Mixed ANOVA für Task Success

Allgemeines Lineares Modell

Innersubjektfaktoren

Maß:	MEASURE_1
Task	Abhängige Variable
1	TaskSuccess 1
2	TaskSuccess 2
3	TaskSuccess 3
4	TaskSuccess 4
5	TaskSuccess 5

Zwischensubjektfaktoren

		Wertelabel	N
Gruppe	1	2D	7
	2	VR	7

Deskriptive Statistiken

	Gruppe	Mittelwert	Std.-Abweichung	N
TaskSuccess1	2D	100,00	,000	7
	VR	85,71	37,796	7
	Gesamt	92,86	26,726	14
TaskSuccess2	2D	100,00	,000	7
	VR	57,14	53,452	7
	Gesamt	78,57	42,582	14
TaskSuccess3	2D	85,71	37,796	7
	VR	85,71	37,796	7
	Gesamt	85,71	36,314	14
TaskSuccess4	2D	100,00	,000	7
	VR	85,71	37,796	7
	Gesamt	92,86	26,726	14
TaskSuccess5	2D	100,00	,000	7
	VR	100,00	,000	7
	Gesamt	100,00	,000	14

Multivariate Tests[a]

Effekt		Wert	F	Hypothese df	Fehler df	Sig.	Partielles Eta-Quadrat
Task	Pillai-Spur	,698	5,211[b]	4,000	9,000	,019	,698
	Wilks-Lambda	,302	5,211[b]	4,000	9,000	,019	,698
	Hotelling-Spur	2,316	5,211[b]	4,000	9,000	,019	,698
	Größte charakteristische Wurzel nach Roy	2,316	5,211[b]	4,000	9,000	,019	,698
Task * GRUPPE	Pillai-Spur	,612	3,553[b]	4,000	9,000	,053	,612
	Wilks-Lambda	,388	3,553[b]	4,000	9,000	,053	,612
	Hotelling-Spur	1,579	3,553[b]	4,000	9,000	,053	,612
	Größte charakteristische Wurzel nach Roy	1,579	3,553[b]	4,000	9,000	,053	,612

a. Design: Konstanter Term + GRUPPE
Innersubjektdesign: Task

b. Exakte Statistik

Mauchly-Test auf Sphärizität[a]

Maß: MEASURE_1

					Epsilon[b]		
Innersubjekteffekt	Mauchly-W	Approx. Chi-Quadrat	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Untergrenze
Task	,112	22,786	9	,007	,703	1,000	,250

Prüft die Nullhypothese, daß sich die Fehlerkovarianz-Matrix der orthonormalisierten transformierten abhängigen Variablen proportional zur Einheitsmatrix verhält.

a. Design: Konstanter Term + GRUPPE
Innersubjektdesign: Task

b. Kann zum Korrigieren der Freiheitsgrade für die gemittelten Signifikanztests verwendet werden. In der Tabelle mit den Tests der Effekte innerhalb der Subjekte werden korrigierte Tests angezeigt.

Tests der Innersubjekteffekte

Maß: MEASURE_1

Quelle		Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Task	Sphärizität angenommen	3714,286	4	928,571	,929	,455	,072
	Greenhouse-Geisser	3714,286	2,810	1321,793	,929	,433	,072
	Huynh-Feldt	3714,286	4,000	928,571	,929	,455	,072
	Untergrenze	3714,286	1,000	3714,286	,929	,354	,072
Task * GRUPPE	Sphärizität angenommen	4285,714	4	1071,429	1,071	,381	,082
	Greenhouse-Geisser	4285,714	2,810	1525,146	1,071	,371	,082
	Huynh-Feldt	4285,714	4,000	1071,429	1,071	,381	,082
	Untergrenze	4285,714	1,000	4285,714	1,071	,321	,082
Fehler(Task)	Sphärizität angenommen	48000,000	48	1000,000			
	Greenhouse-Geisser	48000,000	33,720	1423,469			
	Huynh-Feldt	48000,000	48,000	1000,000			
	Untergrenze	48000,000	12,000	4000,000			

Tests der Innersubjektkontraste

Maß: MEASURE_1

Quelle	Task	Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Task	Linear	1142,857	1	1142,857	2,400	,147	,167
	Quadratisch	1836,735	1	1836,735	2,348	,151	,164
	Kubisch	642,857	1	642,857	,491	,497	,039
	Ordnung 4	91,837	1	91,837	,064	,804	,005
Task * GRUPPE	Linear	1142,857	1	1142,857	2,400	,147	,167
	Quadratisch	204,082	1	204,082	,261	,619	,021
	Kubisch	642,857	1	642,857	,491	,497	,039
	Ordnung 4	2295,918	1	2295,918	1,603	,229	,118
Fehler(Task)	Linear	5714,286	12	476,190			
	Quadratisch	9387,755	12	782,313			
	Kubisch	15714,286	12	1309,524			
	Ordnung 4	17183,673	12	1431,973			

Tests der Zwischensubjekteffekte

Maß: MEASURE_1
Transformierte Variable: Mittel

Quelle	Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Konstanter Term	567000,000	1	567000,000	1984,500	,000	,994
GRUPPE	3571,429	1	3571,429	12,500	,004	,510
Fehler	3428,571	12	285,714			

Mixed ANOVA für Time

Allgemeines Lineares Modell

Innersubjektfaktoren

Maß:	MEASURE_1
Task	Abhängige Variable
1	Time1
2	Time2
3	Time3
4	Time4
5	Time5

Zwischensubjektfaktoren

		Wertelabel	N
Gruppe	1	2D	7
	2	VR	7

Deskriptive Statistiken

	Gruppe	Mittelwert	Std.-Abweichung	N
Time1	2D	38,14	22,169	7
	VR	55,43	40,689	7
	Gesamt	46,79	32,732	14
Time2	2D	67,29	25,947	7
	VR	131,14	106,656	7
	Gesamt	99,21	81,601	14
Time3	2D	78,43	73,907	7
	VR	288,43	95,061	7
	Gesamt	183,43	136,253	14
Time4	2D	21,71	21,274	7
	VR	33,00	63,235	7
	Gesamt	27,36	45,702	14
Time5	2D	6,86	5,178	7
	VR	6,00	,816	7
	Gesamt	6,43	3,589	14

Multivariate Tests[a]

Effekt		Wert	F	Hypothese df	Fehler df	Sig.	Partielles Eta-Quadrat
Task	Pillai-Spur	,888	17,917[b]	4,000	9,000	,000	,888
	Wilks-Lambda	,112	17,917[b]	4,000	9,000	,000	,888
	Hotelling-Spur	7,963	17,917[b]	4,000	9,000	,000	,888
	Größte charakteristische Wurzel nach Roy	7,963	17,917[b]	4,000	9,000	,000	,888
Task * GRUPPE	Pillai-Spur	,707	5,432[b]	4,000	9,000	,017	,707
	Wilks-Lambda	,293	5,432[b]	4,000	9,000	,017	,707
	Hotelling-Spur	2,414	5,432[b]	4,000	9,000	,017	,707
	Größte charakteristische Wurzel nach Roy	2,414	5,432[b]	4,000	9,000	,017	,707

a. Design: Konstanter Term + GRUPPE
Innersubjektdesign: Task

b. Exakte Statistik

Mauchly-Test auf Sphärizität[a]

Maß: MEASURE_1

					Epsilon[b]		
Innersubjekteffekt	Mauchly-W	Approx. Chi-Quadrat	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Untergrenze
Task	,144	20,161	9	,018	,547	,729	,250

Prüft die Nullhypothese, daß sich die Fehlerkovarianz-Matrix der orthonormalisierten transformierten abhängigen Variablen proportional zur Einheitsmatrix verhält.

a. Design: Konstanter Term + GRUPPE
Innersubjektdesign: Task

b. Kann zum Korrigieren der Freiheitsgrade für die gemittelten Signifikanztests verwendet werden. In der Tabelle mit den Tests der Effekte innerhalb der Subjekte werden korrigierte Tests angezeigt.

Tests der Innersubjekteffekte

Maß: MEASURE_1

Quelle		Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Task	Sphärizität angenommen	281165,286	4	70291,321	21,799	,000	,645
	Greenhouse-Geisser	281165,286	2,187	128583,089	21,799	,000	,645
	Huynh-Feldt	281165,286	2,916	96430,835	21,799	,000	,645
	Untergrenze	281165,286	1,000	281165,286	21,799	,001	,645
Task * GRUPPE	Sphärizität angenommen	106454,486	4	26613,621	8,254	,000	,408
	Greenhouse-Geisser	106454,486	2,187	48683,985	8,254	,001	,408
	Huynh-Feldt	106454,486	2,916	36510,535	8,254	,000	,408
	Untergrenze	106454,486	1,000	106454,486	8,254	,014	,408
Fehler(Task)	Sphärizität angenommen	154776,229	48	3224,505			
	Greenhouse-Geisser	154776,229	26,240	5898,549			
	Huynh-Feldt	154776,229	34,989	4423,614			
	Untergrenze	154776,229	12,000	12898,019			

Tests der Innersubjektkontraste

Maß: MEASURE_1

Quelle	Task	Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Task	Linear	32589,257	1	32589,257	27,640	,000	,697
	Quadratisch	149769,000	1	149769,000	75,175	,000	,862
	Kubisch	14955,779	1	14955,779	4,269	,061	,262
	Ordnung 4	83851,250	1	83851,250	13,474	,003	,529
Task * GRUPPE	Linear	2763,457	1	2763,457	2,344	,152	,163
	Quadratisch	53427,020	1	53427,020	26,817	,000	,691
	Kubisch	2649,150	1	2649,150	,756	,402	,059
	Ordnung 4	47614,858	1	47614,858	7,651	,017	,389
Fehler(Task)	Linear	14148,886	12	1179,074			
	Quadratisch	23907,408	12	1992,284			
	Kubisch	42039,971	12	3503,331			
	Ordnung 4	74679,963	12	6223,330			

Tests der Zwischensubjekteffekte

Maß: MEASURE_1
Transformierte Variable: Mittel

Quelle	Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Konstanter Term	369388,929	1	369388,929	100,141	,000	,893
GRUPPE	63661,729	1	63661,729	17,259	,001	,590
Fehler	44264,343	12	3688,695			

Mixed ANOVA für ASQ

Allgemeines Lineares Modell

Innersubjektfaktoren

Maß:	MEASURE_1
Task	Abhängige Variable
1	ASQ1_ges
2	ASQ2_ges
3	ASQ3_ges
4	ASQ4_ges
5	ASQ5_ges

Zwischensubjektfaktoren

		Wertelabel	N
Gruppe	1	2D	7
	2	VR	7

Deskriptive Statistiken

	Gruppe	Mittelwert	Std.-Abweichung	N
ASQ1_ges	2D	16,14	3,078	7
	VR	16,86	3,237	7
	Gesamt	16,50	3,057	14
ASQ2_ges	2D	13,57	3,994	7
	VR	14,86	3,625	7
	Gesamt	14,21	3,725	14
ASQ3_ges	2D	14,00	3,830	7
	VR	8,86	6,842	7
	Gesamt	11,43	5,958	14
ASQ4_ges	2D	16,71	3,638	7
	VR	18,43	4,392	7
	Gesamt	17,57	3,975	14
ASQ5_ges	2D	16,71	1,799	7
	VR	18,86	2,268	7
	Gesamt	17,79	2,259	14

Multivariate Tests[a]

Effekt		Wert	F	Hypothese df	Fehler df	Sig.	Partielles Eta-Quadrat
Task	Pillai-Spur	,691	5,027[b]	4,000	9,000	,021	,691
	Wilks-Lambda	,309	5,027[b]	4,000	9,000	,021	,691
	Hotelling-Spur	2,234	5,027[b]	4,000	9,000	,021	,691
	Größte charakteristische Wurzel nach Roy	2,234	5,027[b]	4,000	9,000	,021	,691
Task * GRUPPE	Pillai-Spur	,455	1,875[b]	4,000	9,000	,199	,455
	Wilks-Lambda	,545	1,875[b]	4,000	9,000	,199	,455
	Hotelling-Spur	,833	1,875[b]	4,000	9,000	,199	,455
	Größte charakteristische Wurzel nach Roy	,833	1,875[b]	4,000	9,000	,199	,455

a. Design: Konstanter Term + GRUPPE
Innersubjektdesign: Task

b. Exakte Statistik

Mauchly-Test auf Sphärizität[a]

Maß: MEASURE_1

					Epsilon[b]		
Innersubjekteffekt	Mauchly-W	Approx. Chi-Quadrat	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Untergrenze
Task	,338	11,284	9	,262	,682	,976	,250

Prüft die Nullhypothese, daß sich die Fehlerkovarianz-Matrix der orthonormalisierten transformierten abhängigen Variablen proportional zur Einheitsmatrix verhält.

a. Design: Konstanter Term + GRUPPE
Innersubjektdesign: Task

b. Kann zum Korrigieren der Freiheitsgrade für die gemittelten Signifikanztests verwendet werden. In der Tabelle mit den Tests der Effekte innerhalb der Subjekte werden korrigierte Tests angezeigt.

Tests der Innersubjekteffekte

Maß: MEASURE_1

Quelle		Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Task	Sphärizität angenommen	402,429	4	100,607	9,615	,000	,445
	Greenhouse-Geisser	402,429	2,728	147,535	9,615	,000	,445
	Huynh-Feldt	402,429	3,903	103,114	9,615	,000	,445
	Untergrenze	402,429	1,000	402,429	9,615	,009	,445
Task * GRUPPE	Sphärizität angenommen	126,143	4	31,536	3,014	,027	,201
	Greenhouse-Geisser	126,143	2,728	46,245	3,014	,048	,201
	Huynh-Feldt	126,143	3,903	32,321	3,014	,028	,201
	Untergrenze	126,143	1,000	126,143	3,014	,108	,201
Fehler(Task)	Sphärizität angenommen	502,229	48	10,463			
	Greenhouse-Geisser	502,229	32,732	15,344			
	Huynh-Feldt	502,229	46,833	10,724			
	Untergrenze	502,229	12,000	41,852			

Tests der Innersubjektkontraste

Maß: MEASURE_1

Quelle	Task	Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Task	Linear	49,207	1	49,207	6,261	,028	,343
	Quadratisch	194,005	1	194,005	22,120	,001	,648
	Kubisch	41,257	1	41,257	4,274	,061	,263
	Ordnung 4	117,959	1	117,959	7,576	,018	,387
Task * GRUPPE	Linear	3,779	1	3,779	,481	,501	,039
	Quadratisch	42,250	1	42,250	4,817	,049	,286
	Kubisch	,114	1	,114	,012	,915	,001
	Ordnung 4	80,000	1	80,000	5,138	,043	,300
Fehler(Task)	Linear	94,314	12	7,860			
	Quadratisch	105,245	12	8,770			
	Kubisch	115,829	12	9,652			
	Ordnung 4	186,841	12	15,570			

Tests der Zwischensubjekteffekte

Maß: MEASURE_1
Transformierte Variable: Mittel

Quelle	Quadratsumme vom Typ III	df	Mittel der Quadrate	F	Sig.	Partielles Eta-Quadrat
Konstanter Term	16817,500	1	16817,500	496,650	,000	,976
GRUPPE	,357	1	,357	,011	,920	,001
Fehler	406,343	12	33,862			

Erklärung

Ich erkläre hiermit gemäß § 17 Abs. 2 APO, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum

Unterschrift