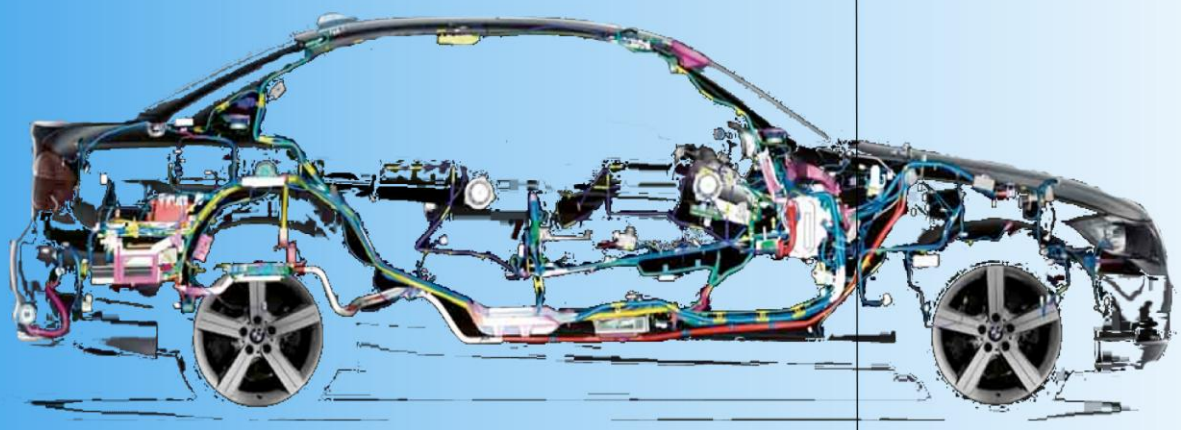
Abschlussbericht



Projektgruppe

VASC

Visual Analytics on Surface Computers

Inhaltsangabe

Abbildungen

Kapitel 1

Einleitung

Dies ist der Abschlussbericht der Projektgruppe *Visual Analytics on Surface Computer* (VASC) an der Carl von Ossietzky Universität Oldenburg in Zusammenarbeit mit dem OFFIS Institut für Informatik und der BMW Group. Eine Projektgruppe soll den Studierenden vermitteln wie ein IT-Projekt mit einer längeren Laufzeit durchgeführt wird. Dabei wird neben der Aneignung von konkreten Fachkenntnissen auch großer Wert auf die Vermittlung von Methoden- und Sozialkompetenz gelegt. Alle Studierenden müssen selbstständig aber als Teil eines Teams zur bestmöglichen Erreichung des Projektziels beitragen.

Die Teilnehmer der Projektgruppe VASC waren:
Johannes Cordes, Volker Gollücke, Daniel Häuser, Raphael Holtmann, Jan Kirchhoff, Andreas Löcken, Mark Phlippen, Marcus Rehnen, Klaas Schmidt, Sebastian Vehring.
Die Projektgruppe stand unter der Zuständigkeit von Prof. Dr. Dr. h.c. H.-Jürgen Appelrath wurde von Dipl. Inf. Stefan Flöring und Dipl. Inf. Tobias Hesselmann betreut.

Der Projektzeitraum erstreckte sich über das Sommersemester 2010 und das Wintersemester 2010/2011.

Motivation

Die zentrale Motivation für die Durchführung einer Projektgruppe in diesem Themengebiet besteht darin, dass in der heutigen Zeit häufig weder rein automatische noch rein manuelle Datenverarbeitungsmechanismen ausreichen, um große Datenmengen in akzeptabler Zeit analysieren und auswerten zu können. Die visuelle Analyse macht sich die Vorteile beider Techniken zu Nutze, in dem sie die intuitive Gabe des Menschen, in großen Datenmengen schnell Muster erkennen zu können, mit der Rechenleistung eines Computers kombiniert. So können in kurzer Zeit Daten aggregiert werden und auf Muster und Besonderheiten untersucht werden, die algorithmisch nur schwer zu

identifizieren sind.[Maz09]

Der Einsatz von Surface Computern kann die visuelle Analyse noch weiter unterstützen. Dies wird dadurch erreicht, dass der Mensch die Daten durch die Berührungssteuerung direkt manipulieren kann. Der Rechner hat hierbei die Aufgabe, die Daten in passender Art zu visualisieren, damit der Mensch sie leichter analysieren kann.

Dies führt zu einer Symbiose aus menschlichem Gehirn und digitalem Prozessor.

Aufbau des Berichts

Dieser Abschlussbericht gliedert sich in fünf Bereiche (**I** - **V**).
Unter **I** befinden sich die Grundlagen dieses Projektes. Es handelt sich dabei um überarbeitete Fassungen der zu Beginn des Projektes angefertigten Seminararbeiten. In diesen Arbeiten wurden grundlegende Themen im Bereich der visuellen Analyse, des Arbeitens mit Surface Computern, sowie der Anwendungsentwicklung für diese behandelt. Darüber hinaus wurden weiterführende Themen behandelt, die sich mit der von BMW zur Verfügung gestellten Datengrundlage des Projektes befassen. Außerdem wurden grundlegende Aspekte des Projektmanagements betrachtet.

In Abschnitt **II** wird die Organisation des Projektes behandelt. Dabei wird erläutert, welches grundlegendes Vorgehensmodell die Projektgruppe verfolgt hat und welcher Hilfsmittel und Methoden sich das Projektmanagement bedient hat.

Abschnitt **III** beschreibt den Aufbau und die Entstehung des von VASC entwickelten Softwareprototypen *TOAD*. Dabei wird darauf eingegangen, wie der Kontakt mit dem Projektpartner BMW organisiert wurde und wie daraus die Anforderungen an das Programm entstanden sind. Im nächsten Schritt wird aufgezeigt, wie - entsprechend dem Vorgehensmodell *SCiVA* - aus den Anforderungen die einzelnen Anwendungsfälle abgeleitet wurden und wie diese schließlich implementiert wurden.

Abschnitt **IV** beinhaltet eine kurze Zusammenfassung, sowie einen Ausblick darüber, inwiefern der Prototyp durch weitergehende Arbeiten ergänzt werden könnte. Abschnitt **V** beinhaltet den Anhang mit Literaturverzeichnis.

Teil I

Grundlagen

Kapitel 2

Visual Analytics

In der heutigen Zeit, wo Speicherkapazitäten und damit auch das Datenaufkommen rasant ansteigen, werden verstärkt neue Analysewerkzeuge benötigt. Denn die bekannten und vorher auch durchaus erfolgreichen Data-Mining-Verfahren scheitern an dieser komplexen und enormen Datenmengen. Um in dieser Flut von Daten die häufig versteckten Informationen zu entdecken, hat sich die visuelle Analyse als äußerst erfolgreiche Methode herausgestellt. Bei dieser Analysetechnik wird der Mensch in den Explorationsprozess mit eingebunden. Mit seinem intuitiven Vorgehen, seinem Allgemeinwissen und seiner Fähigkeit zu kombinieren, ist er der entscheidende Faktor zur effektiveren Analyse eines großen Datenaufkommens. Dabei kann der Analyst den Analyseablauf steuern, Teilergebnisse beurteilen und daraufhin entscheiden, ob er in der Richtung weiter exploriert oder die bisherigen Ergebnisse verwirft und einen neuen Weg einschlägt. Um diesen neuen Weg der Datenanalyse noch effektiver zu gestalten, werden gleich mehrere Personen in den Prozess mit eingebunden. Somit kam die Idee, den Analyseprozess mit mehreren Analysten an einem Multi-Touch Surface-Computer durchzuführen, um die kollaborative Arbeit zu stärken. Durch die Auffassungsgabe des Menschen und die Zusammenarbeit an dem Multi-Touch Eingabegerät, können somit schneller effektivere Erkenntnisse aus den Visualisierungen gewonnen werden.

2.1 Grundlagen

In der Literatur gibt es zahlreiche Definitionen, die hier nicht alle aufgeführt werden sollen. Jedoch definiert Prof. Dr.-Ing. habil. Heidrun Schumann die visuelle Analyse sehr passend:

> ”Visual Analytics bezeichnet die Verknüpfung von automatischen und interaktiv visuellen Methoden für eine effektive Exploration komplexer heterogener Informationsmengen. Ziel ist es dabei, die Leistungsfähigkeiten moderner Computer und Algorithmen mit den enormen Leistungsfähigkeiten des menschlichen visuellen Systems zu verbinden, um eine umfassende

> Analyse komplexer Datenmengen zu unterstützen, unbekannte in den Daten verborgene Muster aufzudecken und Trends abzuleiten. Dabei geht es um eine umfassende Kombination von Ansätzen ganz unterschiedlicher Disziplinen, was gleichzeitig auch die größte Herausforderung darstellt." [Sch07b]

Mit anderen Worten kann gesagt werden, dass Visual Analytics ein sich immer wiederholender Prozess ist, der die Informationsgewinnung, Datenvorbehandlung, Wissenspräsentation, Interaktion und die Entscheidungsfindung einschließt. Das ultimative Ziel der visuellen Analyse ist es dabei, Einblicke in ein bestehendes Problem zu erlangen, welches von einer sehr großen Anzahl von Daten verschiedener Quellen beschrieben wird. Damit dieses Ziel erreicht werden kann, werden die Stärken von Computern mit denen des Menschen kombiniert [KMS+08]. Ferner ist Visual Analytics ein multidisziplinarisches Wissenschaftsfeld. Dies wird in Abbildung 2.1 detailliert dargestellt.

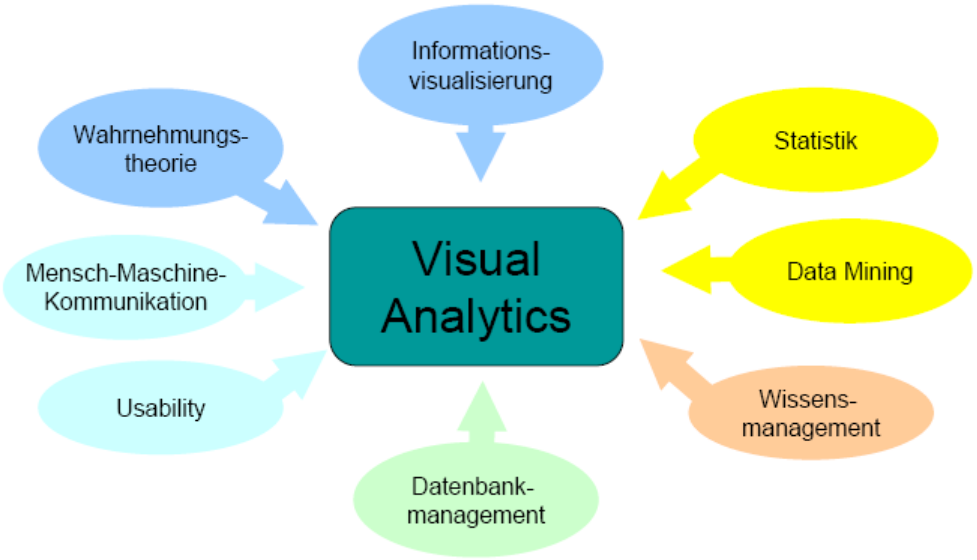


Abbildung 2.1: Umfeld der Visual-Analytics [TD09]

Die visuelle Analyse ist ein wachsendes Forschungsgebiet, in dem neue Technologien für die bestmögliche Nutzung von sehr großen Datenmengen entwickelt werden. Die grundlegende Idee dabei ist, die Stärken von intelligenter, automatischer Datenanalyse mit den menschlichen Fähigkeiten zur visuellen Wahrnehmung und kognitiver Analyse zu verbinden. So gesehen ist die visuelle Analyse eine Kombination aus Informationsvisualisierung, Statistik, Data Mining, Business Intelligence und anderen Gebieten.

Die klassischen Data-Mining Verfahren scheitern heutzutage an der großen Menge an Daten. Diese stellen zumeist einen riesigen Dschungel aus Datenquellen und -formaten dar. Außerdem ist die Präsentation der Daten oft sehr unübersichtlich z.B.

in Tabellen, wo Abweichungen oder Zusammenhänge nur schwer zu erkennen sind. Oft wird dieser Berg an Informationen als Hindernis oder Ballast angesehen, doch dem ist nicht so. Die Datensammlung und -analyse haben immer mehr an Bedeutung in verschiedensten Unternehmensbereichen gewonnen [Ins11]. Daher wählt man bei der visuellen Datenexploration einen anderen Ansatz: Große Datenmengen sind „ein Fall für die Statistik“. Durch die Darstellung in Diagrammen werden Strukturen und Muster sichtbar sowie Abweichungen von Sollwerten, die in kürzester Zeit vom menschlichen Auge erkannt werden können [Int11]. Die Kombination aus den menschlichen Fähigkeiten -Flexibilität, Kreativität und das Allgemeinverständnis- mit den enormen Speicherkapazitäten und Rechenleistungen moderner Computersysteme, machen diese Analysetechnik sehr erfolgversprechend. Dadurch dass der Mensch direkt an dem Explorationsprozess beteiligt ist, können die Explorationsziele bei Bedarf verändert und angepasst werden. Die Hauptvorteile der Einbindung des Menschen in den Analyseprozess im Vergleich zu den vollautomatischen Verfahren aus der Statistik beziehungsweise künstlichen Intelligenz sind: Der visuelle Analyseprozess kann stark inhomogene und vertauschte Daten verarbeiten. Der Analyst benötigt keine Kenntnisse von komplexen mathematischen oder statistischen Algorithmen und Parametern. Deshalb kann die Datenexploration von Nicht-Spezialisten durchgeführt werden. Die visuelle Analyse ist in vielen Fällen eine einfachere Exploration der Daten, die oft auch bessere Ergebnisse erzielt. In Verbindung mit automatischen Algorithmen ist die visuelle Datenexploration ein unentbehrliches Verfahren zur Exploration wichtiger Informationen aus großen Datenbanken [Mag11].

Bei dem visuellen Analyse-Prozess werden automatische und visuelle Analysemethoden eng gekoppelt mit der menschlichen Interaktion, um Wissen aus einer großen Menge von Daten zu gewinnen. In der folgenden Abbildung 2.2 wird die visuelle Datenexploration mit ihren verschiedenen Phasen (durch Ovale dargestellt) und deren Übergänge (Pfeile) dargestellt.

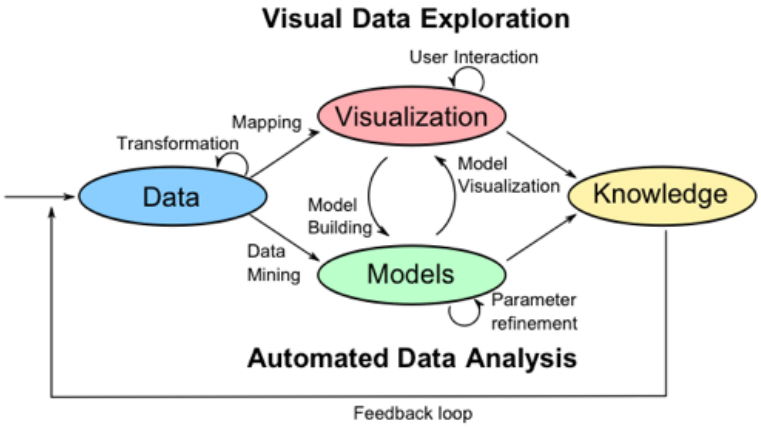


Abbildung 2.2: Visual-Analytics Process [Vis11b]

In vielen Anwendungsszenarien müssen die Daten vor dem Analysieren aus heterogenen Datenquellen integriert werden. Der erste Schritt bei einer visuellen oder automatisierten Analyse-Methode ist oft die Vorbereitung und Transformation der Daten in verschiedene Darstellungen für die weitere Exploration. Des Weiteren gehören zur Vorverarbeitung der Daten: das Reinigen, die Normalisierung, die Gruppierung oder die Integration von heterogenen Datenquellen. Nach der Bearbeitung der Daten kann der Analytiker wählen zwischen der visuellen oder der automatischen Datenanalyse. Wenn eine automatisierte Analyse zuerst verwendet wird, werden Data-Mining Methoden angewandt, um Modelle der ursprünglichen Daten zu generieren. Nachdem ein Modell geschaffen wurde, muss der Analyst dieses bewerten und verfeinern, was am besten durch Interaktion mit den Daten erfolgen kann. Der Analyst kann mit automatischen Methoden durch eine Visualisierung explorieren, indem er durch Änderung der Parameter oder Auswahl anderer Analyse-Algorithmen mit ihr interagiert. Der Wechsel zwischen visuellen und automatischen Verfahren ist charakteristisch für den visuellen Analyse-Prozess und führt zu einer kontinuierlichen Weiterentwicklung und Überprüfung der vorläufigen Ergebnisse. Irreführende Ergebnisse in einem Zwischenschritt, können somit in einem frühen Stadium entdeckt werden. Dies führt zu besseren und vertrauenswürdigeren Resultaten. Falls der Analyst zuerst eine visuelle Exploration durchführt, hat er die generierten Hypothesen durch eine automatisierte Analyse zu bestätigen. Durch die Benutzerinteraktion mit der Visualisierung können aufschlussreiche Informationen offenbart werden, zum Beispiel durch Zoomen in verschiedene Datenbereiche oder durch die Berücksichtigung verschiedener visueller Sichten auf die Daten. Entdeckungen in der Visualisierung können zur Modellbildung in der automatischen Analyse führen. Somit kann mit dem visuellen Analyse-Prozess, also der automatischen Analyse sowie der vorangegangenen Interaktion zwischen Visualisierung und menschlichen Analysten, Wissen von der Datenvisualisierung gewonnen werden [Vis11a].

2.2 Nutzen

Für eine effektive Datenanalyse ist es wichtig, den Menschen in die Datenexploration mit einzubinden, um somit die Flexibilität, die Kreativität und das Allgemeinwissen des Menschen mit den Vorzügen heutiger Computer zu kombinieren. [KW07] Dies soll in Abbildung 2.3 weiter verdeutlicht werden.

Die visuelle Analyse hat immer dann einen sehr hohen Nutzen, wenn über die Daten wenig bekannt ist, oder die Ziele der Exploration noch nicht festgelegt sind. Da der Mensch am Explorationsprozess beteiligt ist, können die Ziele während der Exploration nach Bedarf ermittelt oder angepasst werden. [KW07]

Die visuelle Analyse kann auch als ein Prozess zur Generierung von Hypothesen angesehen werden: Die Visualisierung soll dem Nutzer ermöglichen, einen Einblick in

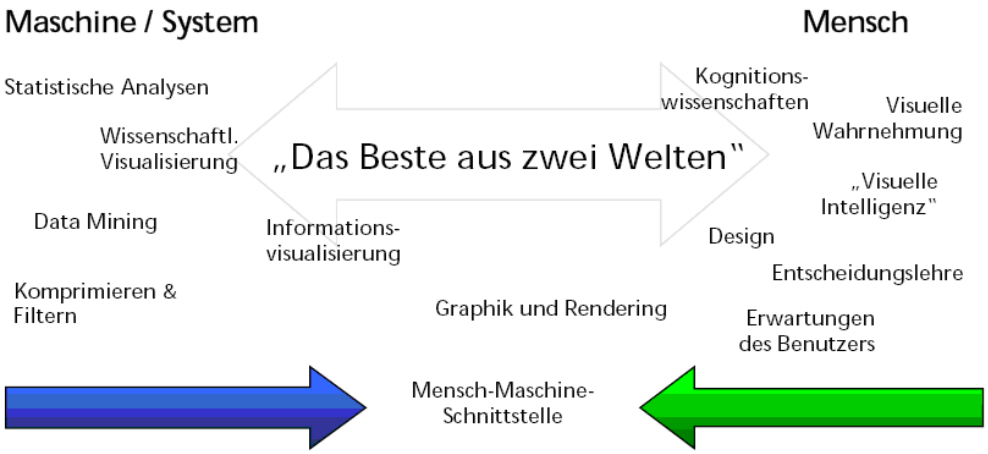


Abbildung 2.3: Arbeitsteilung zwischen Mensch und Computer [IGD07]

die Daten zu erhalten und dadurch neue Hypothesen aufzustellen. Im Anschluss darauf können die Hypothesen auf ihre Wahrheit hin untersucht werden. Dies kann ebenfalls durch Visualisierung geschehen, aber auch durch automatisierte Techniken. [KW07]

Die Vorteile der Einbindung des Menschen in den Datenexplorationsprozess im Vergleich zu vollautomatisierten Analysemethoden sind:

- heterogene und verrauschte Daten können einfach verarbeitet werden.
- die visuelle Datenanalyse ist intuitiv und verlangt dem Nutzer kein komplexes mathematisches Wissen ab.
- die Visualisierung kann einen guten Überblick geben und interessante Bereiche aufzeigen, die für weitere Analysen isoliert betrachtet werden sollten.
- der Weg von den Daten zu einer Entscheidung wird verkürzt. [KW07]

Zusammenfassend kann gesagt werden, dass die visuelle Datenexploration meistens schnellere und vor allem interessantere Ergebnisse liefert. Ferner kann es auch dann eingesetzt werden, wenn andere automatisierte Analysen bereits fehlschlagen. Außerdem kann den Ergebnissen der visuellen Analyse hohes Vertrauen angerechnet werden, da der Mensch die Analyse von Beginn an verfolgt. Diese Fakten führen zu einer hohen Nachfrage von visuellen Analysemethoden in Kombination mit automatisierten Verfahren. [KW07]

2.3 Klassifizierung visueller Analysetechniken

In diesem Abschnitt werden die visuellen Analysetechniken vorgestellt. Diese können durch drei Kriterien klassifiziert werden, die zur Übersicht in Abbildung 2.4 dargestellt sind.

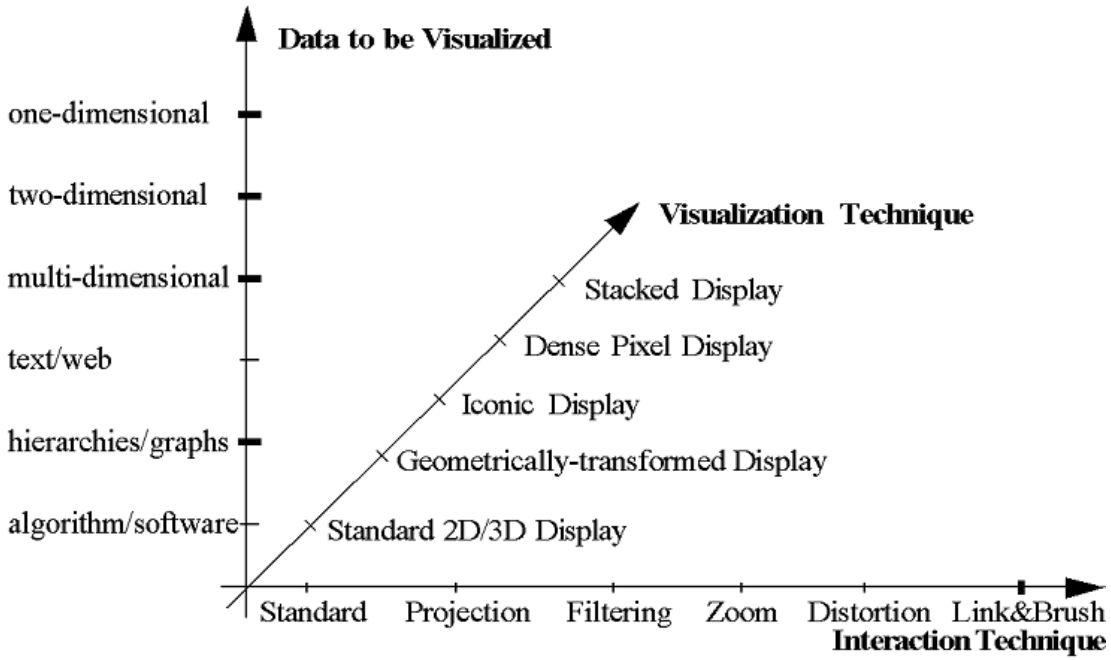


Abbildung 2.4: Klassifizierung der visuellen Analysetechniken [KW07]

2.3.1 Zu visualisierende Datentypen (Data to be Visualized)

Die in der Informationsvisualisierung verwendeten Daten besitzen im Regelfall eine große Anzahl Datensätze. Dabei entspricht jeder einer Beobachtung, Messung oder Transaktion von Datensätzen. Außerdem besitzt ein Datensatz eine Menge verschiedener Dimensionen, die je nach Anwendung stark variieren können [Sch07a].

Eindimensionale Daten (one-dimensional)

Eindimensionale Daten besitzen in der Regel ein gleichbleibendes Attribut. Ein typisches Beispiel sind zeitabhängige Daten, bei denen zu jedem Zeitpunkt genau ein Wert existiert. Dies ist zum Beispiel bei einem Aktienverlauf der Fall [KW07].

Zweidimensionale Daten (two-dimensional)

Zweidimensionale Daten bestehen aus zwei Dimensionen, die jeden Punkt eindeutig beschreiben. Ein typisches Beispiel sind geographische Daten, dessen beiden Dimensionen durch Breiten- und Längengrad charakterisiert werden. Zur Darstellung solcher Daten eignen sich besonders Koordinatensysteme. Auf den ersten Blick scheint diese Visualisierung einfach zu sein, doch häufig gibt es aufgrund der hohen Datenmengen viele Überlappungen, wodurch die Visualisierung nicht hilft, die Daten zu verstehen [KW07].

Multidimensionale Daten (multi-dimensional)

Daten bestehen aus mehr als drei Dimensionen und erlauben daher keine einfache 2D oder 3D Visualisierung. Ein Beispiel für solche Daten sind Tabellen von relationalen Datenbanken, die häufig zehn bis hundert Spalten, also Dimensionen, besitzen. Da keine einfache Abbildung dieser Daten auf den beiden Dimensionen eines Bildschirms möglich ist, werden anspruchsvollere Visualisierungsmöglichkeiten notwendig. Ein Beispiel für eine Visualisierungstechnik, die leicht und verständlich multidimensionale Daten abbilden kann ist die parallele Koordinatentechnik, die in Abbildung 2.5 dargestellt ist. [KW07] Dabei werden mehrere Koordinatensysteme parallel angeordnet. Dort wird ersichtlich, dass weiße, ältere Männer mit vielen Studienjahren und einer hohen Wochenstundenzahl eine hohe Wahrscheinlichkeit haben, ein Einkommen von mehr als 50.000 $ zu erziele [XLS10].

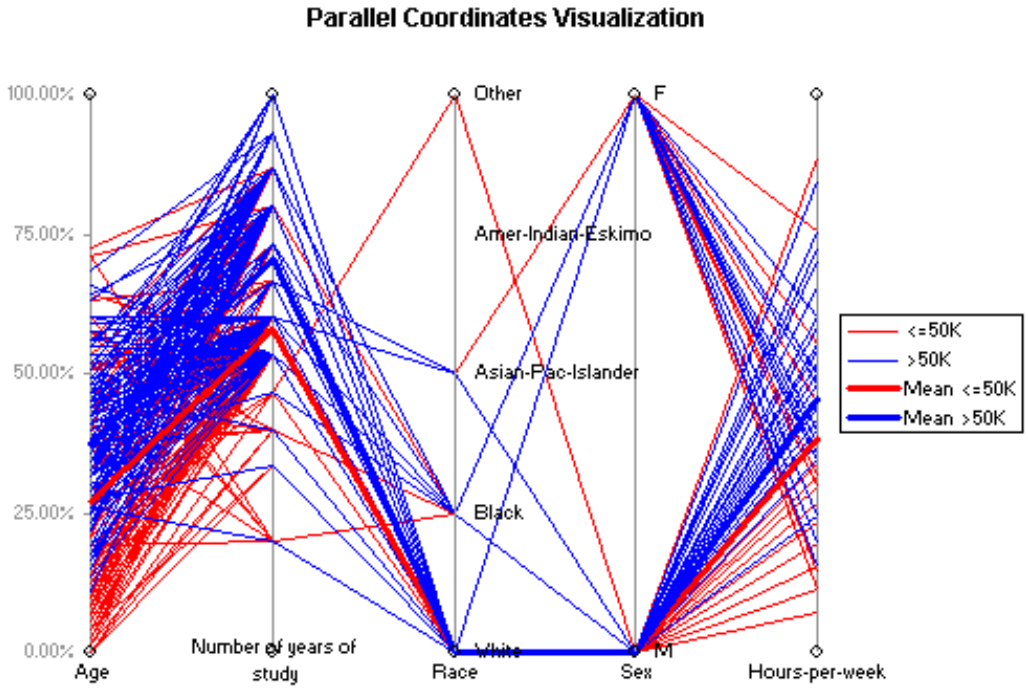


Abbildung 2.5: Parallele Koordinaten [XLS10]

Text und Hypertext (text/web)

Text und Hypertext werden als Datentypen im Zeitalter des World Wide Web immer wichtiger. Diese Datentypen können jedoch nur schwer durch Dimensionen beschrieben werden. Daher sind die Standard-Visualisierungstechniken meist unzureichend. Um diese Datentypen dennoch visualisieren zu können, werden sie in so genannte Beschreibungsvektoren umgewandelt. Ein Beispiel dafür ist das Zählen der Wörter [KW07]. Dies wird in Abbildung 2.6 dargestellt. Textdokumente können als Landschaften dargestellt werden, dessen Berge den Themengebieten entsprechen, die häufig vorkommen [JC05].

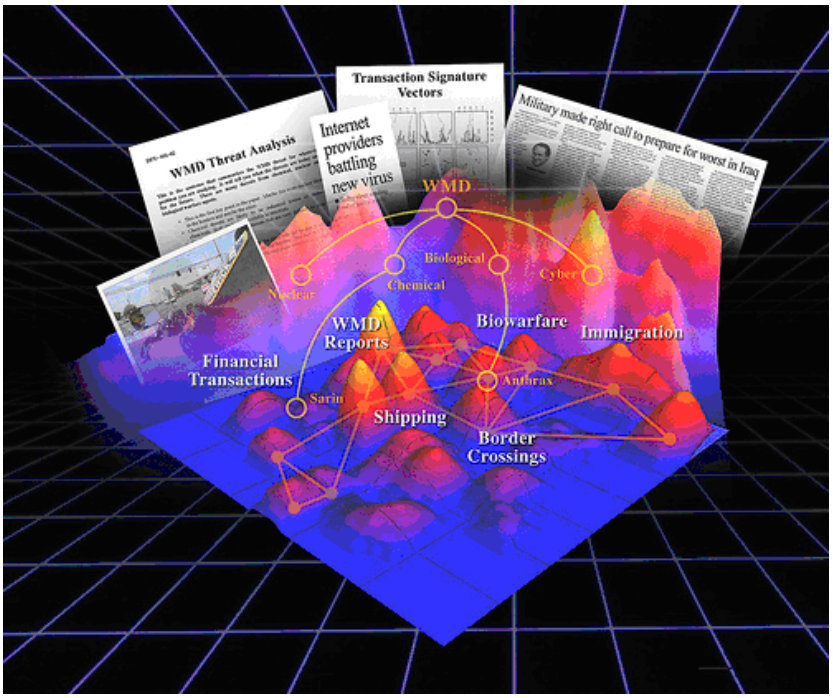


Abbildung 2.6: ThemeView-Visualisierung [Fli07]

Hierarchien und Graphen (hierachies/graphs)

Datensätze besitzen häufig komplexe Beziehungen untereinander. Diese können mit Hilfe von Graphen dargestellt werden. Ein Graph besteht aus Objekten (auch Knoten genannt) sowie aus Verbindungen (auch Kanten genannt) zwischen diesen Objekten [KW07].

Algorithmen und Software (algorithm/software)

Eine weitere Klasse sind Algorithmen und Software. Da die Durchführung großer Softwareprojekte eine große Herausforderung darstellt, ist das Ziel dieser Visualisierung die Unterstützung der Softwareentwickler für ein besseres Verständnis des Codes [KW07].

2.3.2 Visualisierungstechnik (Visualization Technique)

Es gibt eine große Anzahl Visualisierungstechniken. Neben den Standard 2D/3D-Techniken, wie den X-Y-, Balken- oder Liniendiagrammen, gibt es bereits eine Menge anspruchsvollerer Techniken, die im Folgenden vorgestellt werden [KW07].

Geometrische Transformationen (Geometrically-transformed Display)

Geometrische Transformationen haben zum Ziel, interessante Projektionen multidimensionaler Daten zu finden, um sie dann visuell darzustellen. Hierzu gehört zum Beispiel die bereits geschilderte Technik der parallelen Koordinaten, die in Abbildung 2.5 dargestellt wurde [KW07].

Icon-basierte Visualisierungen (Iconic Display)

Die Idee dieser Visualisierungsform ist die Abbildung der Dimensionen eines Datensatzes auf die Merkmale eines Symbols. Als Symbole können zum Beispiel kleine Gesichter oder Sterne genommen werden. In Abbildung 2.7 ist eine Strichmännchen-Visualisierung einer Volkszählung zu sehen.

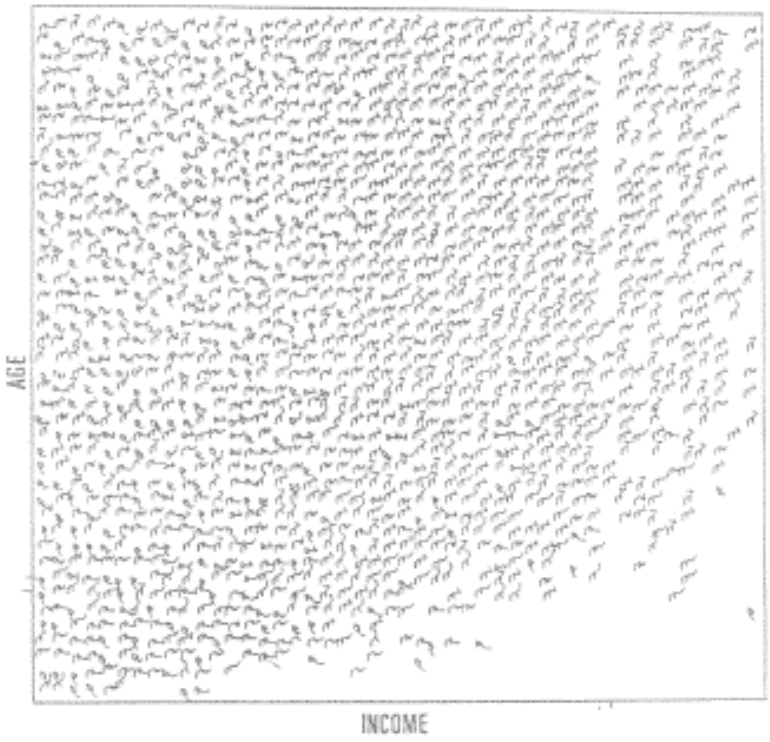


Abbildung 2.7: Strichmännchen-Visualisierung [ST05]

Die Konturen der Strichmännchen beinhalten Daten wie die Schulbildung oder Abstammung. Es fällt auf, dass die Strichmännchen weiter rechts (bedeutet hohes Einkommen) gleichmäßiger gezeichnet sind, was schließen lässt, dass ein hohes Einkommen häufig an einige Parameter gebunden ist. Im linken Bereich sind die Strichmännchen jedoch sehr unterschiedlich [ST05].

Pixel-Visualisierungen (Dense Pixel Display)

Hierbei besteht die Idee darin, jeden Datenwert als einen farbigen Pixel darzustellen. Gruppiert werden die Pixel dabei nach ihrer Dimension. Auf heutigen Bildschirmen können problemlos Millionen von Datenwerten dargestellt werden [KW07].

Geschachtelte Visualisierungen (Stacked Display)

Diese Visualisierungsform stellt Daten in hierarchischer Form dar. Dabei werden die Wertebereiche der Dimensionen ineinander geschachtelt. Ein Beispiel für diese Hierarchien ist die in Abbildung 2.8 dargestellte Treemap. Diese visualisiert große, hierarchisch organisierte Daten, bei denen die Ebenen der Hierarchien durch ineinander geschachtelte Rechtecke dargestellt werden [ST05].

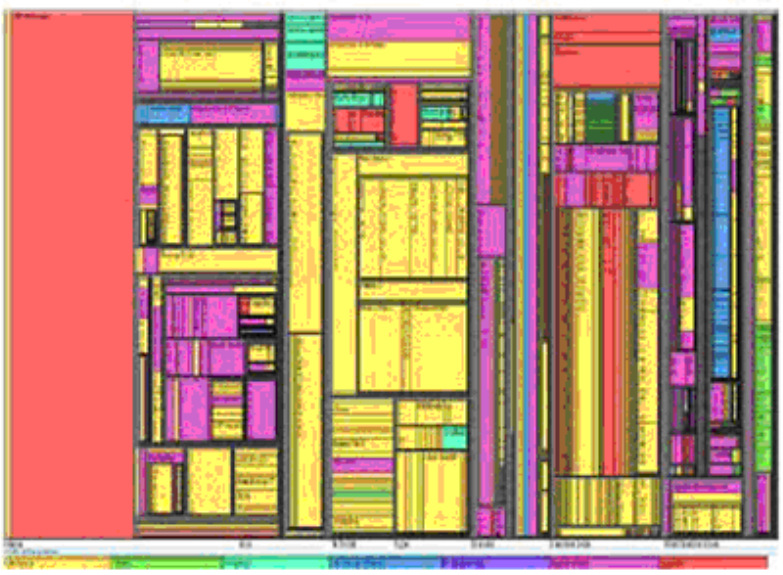


Abbildung 2.8: Treemap-Visualisierung [ST05]

2.3.3 Interaktionstechniken (Interaction Technique)

Um eine effektive Datenexploration zu betreiben sind geeignete Interaktionstechniken unverzichtbar. Diese erlauben dem Nutzer, die Visualisierungen gezielt nach seinen Explorationszielen zu verändern. Außerdem machen sie es möglich, verschiedene Visualisierungstechniken zu kombinieren [KW07].

Dynamische Projektion (Projection)

Die Absicht der dynamischen Projektion ist die dynamische Änderung der Darstellung einer multidimensionalen Datenmenge [KW07].

Interaktive Filterung (Filtering)

Bei der visuellen Exploration sehr großer Daten ist eine interaktive Aufteilung der Daten in Bereiche und eine Auswahl von interessanten Teilmengen wichtig. [KW07]

Interaktives Zooming (Zoom)

Das Zooming ist eine weit verbreitete Technik. Bei großen Datenmengen ist es wichtig, diese in komprimierter Form darzustellen, um einen Überblick zu erhalten. Aber ebenso wichtig ist es, eine variable Auflösung der Darstellung zu erhalten, um weiter in die Daten hineinzublicken. Zooming bedeutet dabei aber nicht, die Daten nur größer darzustellen, sondern mehr Details preiszugeben. Dabei kann sich gegebenenfalls auch die Visualisierung ändern. In einer Gesamtübersicht können zum Beispiel Daten als Pixel dargestellt werden, bis sie ab einem bestimmten Zoomlevel als Symbole visualisiert werden [KW07].

Interaktive Verzerrung (Distortion)

Bei der interaktiven Verzerrungstechnik soll der Überblick über die Daten erhalten bleiben. Dabei soll es möglich sein, einen Teil der Daten genauer zu erforschen, während der Überblick über die Daten zu jedem Zeitpunkt beibehalten wird. In Abbildung 2.9 ist ein solches Beispiel dargestellt [KW07].

Interaktives Linking and Brushing (Link & Brush)

Die Visualisierungstechniken, die wir im vorherigen Abschnitt kennengelernt haben, besitzen, je nach Anwendung, ihre Schwächen und Stärken. Das interaktive Linking and Brushing (Verknüpfung und Einfärbung) zielt auf die Zusammenführung verschiedener Visualisierungstechniken ab, um die Nachteile der einzelnen Darstellungen auszugleichen [KW07].

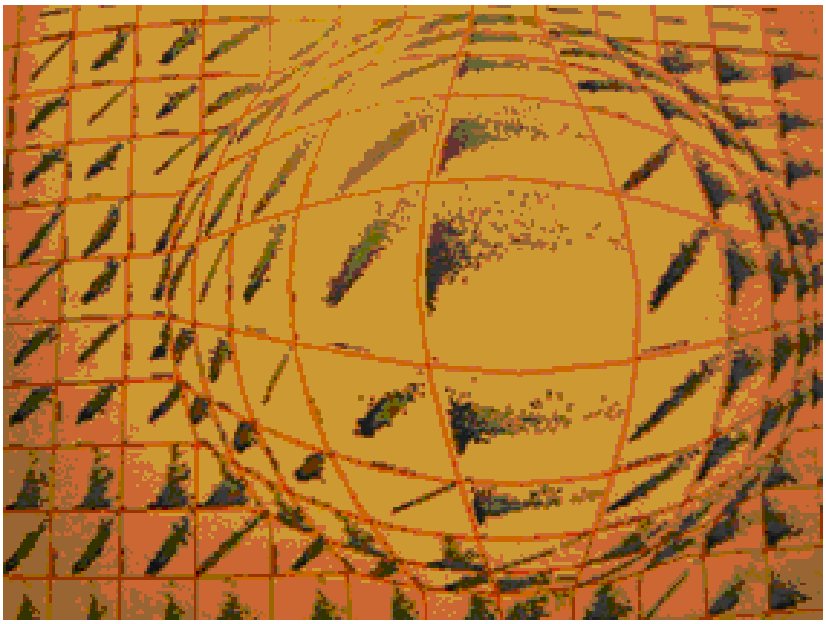


Abbildung 2.9: Distortion einer scatterplot matrix [KW07]

2.4 Visualisierungsdesign

2.4.1 Einleitung

Dieses Kapitel stellt eine Einführung in die projektrelevanten Bereiche der Gestaltung von Informationsvisualisierungen im Kontext der explorativen visuellen Analyse von Daten, der so genannten *Visual Analytics* dar.

Grundlagen zum Kapitel

Zu beachten ist, dass es sich beim Gegenstand der Projektgruppe ausdrücklich um interaktive Visualisierungen handelt. Also werden nicht nur die Möglichkeiten zur Visualisierung der Daten an sich, sondern auch die Methoden zur Manipulation dieser Darstellungen behandelt. Auch ein Modell zur Entwicklung von Visualisierungen und Gütekriterien werden dargestellt.

Die Gestaltung der Benutzeroberfläche der zu erstellenden Anwendung wird an anderer Stelle in diesem Dokument betrachtet.

Im Folgenden werden zunächst die relevanten Begriffe definiert. Anschließend werden verschiedene Möglichkeiten Daten darzustellen aufbereitet und auch Methoden zur Manipulation dieser Darstellungen beschrieben.

Abschließend erfolgt eine Einordnung der dargestellten Erkenntnisse hinsichtlich der Einsetzbarkeit für den Zweck der Projektgruppe.

Begriffsdefinitionen

Visualisierung Eine Visualisierung bezeichnet im Kontext dieser Arbeit grundsätzlich eine optisch erfassbare Darstellung, siehe auch Informationsvisualisierung.

Informationsvisualisierung In dieser Arbeit bezieht sich der Begriff Informationsvisualisierung speziell auf wissenschaftlich-technische Visualisierungen, im Englischen auch *scientific visualisation* genannt, deren Zweck es unter anderem ist, Daten verständlich und analysierbar aufzubereiten, so dass sie zum Beispiel für eine explorative Datenanalyse genutzt werden können.

2.4.2 Informationsvisualisierung und Interaktion

Gestaltung einer interaktiven Visualisierung nach Mazza

Mazza schlägt ein einfaches aber sinnvolles Vorgehensmodell zur Gestaltung einer *Visual Application*, also einer unter Umständen auch interaktiven Visualisierung vor, welches im Folgenden zusammengefasst dargestellt werden soll:
Zunächst soll das zu lösenden Problem definiert werden, in dem die Anforderungen der Nutzer identifiziert werden. Dies soll laut Mazza durch das Beobachten der Nutzer bei ihrer Arbeit geschehen, da so die Anforderungen der Nutzer am besten erfasst werden können.[Maz09, S. 25 f.] Auch hier werden die drei typischen Intentionen von Visualisierungen berücksichtigt:

- Vorhandene Information kommunizieren
- Eine Hypothese beweisen
- Neue Informationen gewinnen

Wenn der letzte Fall vorliegt, kann von einem typischen Anwendungsfall der Visual Analytics ausgegangen werden.[Maz09, S. 25 f.] Weiterhin sollen auch das Vorwissen und sogar die kognitiven Fähigkeiten der Nutzer berücksichtigt werden.

Danach sollen die Ausgangsdaten untersucht werden. Mazza beschränkt sich hier auf eine Kategorisierung der Daten auf ihren Typ, ob es sich also zum Beispiel um quantitative oder ordinale Daten handelt. An dieser Stelle wäre bereits eine weitergehende Untersuchung möglich (zum Beispiel im Hinblick auf Datenformate), ist aber nicht notwendig.

Im dritten Schritt soll die Anzahl der Dimensionen der Daten ermittelt werden, da diese, wie der Datentyp, wesentlich die Auswahl der Visualisierungstechnik beeinflusst. Dabei wird die Anzahl der unabhängigen Attribute der Daten berücksichtigt. Möglich sind hierbei univariate Daten, also Daten bei denen eine Dimension von der anderen abhängt, bi-, tri- oder multivariate Daten, also Daten bei denen zwei oder mehr Dimensionen von den jeweils anderen abhängen.

Ein weiterer wichtiger Untersuchungsschritt zur Findung der optimalen Visualisierungsmethode ist die Untersuchung der Datenstruktur. Es können zum Beispiel lineare oder zeitabhängige Daten vorliegen. Besondere Visualisierungen müssen genutzt werden, wenn geographische oder hierarchische Daten vorliegen.

Zuletzt soll die Art der möglichen Interaktionen festgelegt werden. Im Regelfall sind die Visualisierungen nicht statisch sondern transformierbar oder manipulierbar. In den beiden letzten Fällen können die Daten bzw. die Visualisierungen durch den Benutzer in verschiedenen Aspekten verändert werden. So können einfache Möglichkeiten wie das Vergrößern oder Verkleinern der Darstellung, aber auch komplexere, wie zum Beispiel das Rotieren von dreidimensionalen Darstellungen oder das Austauschen von Dimensionen einer Darstellung realisiert werden.[Maz09, S. 25 f.] Verschiedene Interaktionsmöglichkeiten werden später zusammengefasst dargestellt.

Visualisierungsformen

In der Literatur finden sich zahlreiche Möglichkeiten Informationen konkret einer optischen Darstellung zuzuordnen.
Mazza und Schumann zitieren an dieser Stelle zwei sehr interessante Studien, in denen die Eignung der grafischen Attribute zur genauesten möglichen Darstellung von Eigenschaften bzw. Werten erprobt wurde.[Maz09, S. 20 ff] Das interessanterweise übereinstimmende Ergebnis dieser beiden Studien soll beispielhaft für quantitative

Daten dargestellt werden. Die von den Probanden bewerteten Elemente in abnehmender Reihenfolge ihrer zugeordneten Genauigkeit lauten:

- Position
- Länge
- Winkel (orientation)
- Fläche
- Volumen
- Helligkeit
- Sättigung
- Farbton
- Weitere eher ungeeignete Attribute wie Textur oder Form

[Maz09, S.20ff]

Diese Erkenntnisse können dann genutzt werden um konkrete Visualisierungen zu gestalten. Dabei gibt es quasi unbegrenzte Möglichkeiten, da jeder Typ von Visualisierung in vielen Aspekten im Hinblick auf die verwendeten Daten angepasst werden kann. Einige typische Visualisierungsformen und entsprechende Designhinweise sollen im Folgenden vorgestellt werden.

Punktediagramme Punktediagramme, auch *scatter plots*, sind eine Möglichkeit Daten sehr genau wahrnehmbar darzustellen (siehe oben). Typisches Beispiel ist eine zweidimensionale Darstellung, mit der univariate Daten dargestellt werden können. Auf der waagerechten Achse, der Abszisse, der Zeitverlauf der Daten dargestellt wird, und auf der horizontalen Achse der zugeordnete Wert, zum Beispiel ein Messwert.

Im Kontext des Visualisierungsdesign ist diese Darstellung als einfach aber effektiv zu bewerten, da es den betrachtenden Personen sehr schnell möglich ist bestimmte Muster bzw. Korrelationen zu erkennen. Auch die Art und die Stärke der Korrelation können sehr schnell erkannt werden.

Liniendiagramme Liniendiagramme, bzw. *Kurvendiagramme* nutzen durch die Verwendung der Darstellungsoptionen Position und Länge die besten Möglichkeiten zur Darstellung von quantitativen Daten. Die Werte werden wie im ersten Beispiel als Punkte relativ zu den zwei Achsen abgebildet. Die einzelnen Punkte werden wiederum mit Linien verbunden. In dieser Visualisierungsform können mehrere Datenreihen gleichzeitig visualisiert werden, wobei die Punkte und Linien durch unterschiedliche Formatierung, wie zum Beispiel Färbung unterscheidbar gehalten werden können. Die so dargestellten Daten sind sehr gut vergleichbar. Allerdings empfehlen Schumann und Müller nicht mehr als vier Datenreihen gleichzeitig darzustellen.

Kreisdiagramme Kreisdiagramme oder *pie charts*, sind eine der simpelsten Darstellungsformen. Die einzelnen Werte werden durch die Fläche der Segmente dargestellt. Dies macht schon eine bedeutende Einschränkung deutlich: Sie eignet sich nur, wenn eindimensionale, quantitative Daten dargestellt werden sollen. Außerdem muss das Kreisdiagramm in ausreichender Größe dargestellt werden, damit die Flächen noch optisch miteinander verglichen werden können. Weiter eignet es sich nur wenn bis zu etwa 15 Werte dargestellt werden sollen. Um die Wahrnehmbarkeit zu verbessern können einzelne Segmente farblich oder durch herauslösen aus dem Kreis hervorgehoben werden.

Spiechart Das so genannte Spiechart (s. Abb. 2.10) wurde von Dror G. Feitelson entwickelt um einige der oben genannten Nachteile der einfachen Piecharts aufzuheben. Es handelt sich vereinfacht gesagt um ein zweidimensionales Kreisdiagramm. Die erste Dimension der Daten wird wie beim Piechart durch den Winkel der Segmente dargestellt. Die zweite Dimension wird durch die Fläche, bzw. den Radius der Segmente abgebildet.[Fei03]

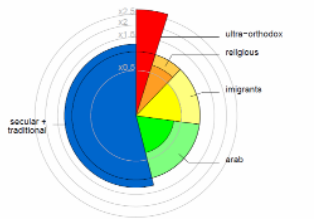


Abbildung 2.10: Spie-Chart aus Feitelson 2003, S.4

Data-Lens-Methode Die Data-Lens-Methode eignet sich im Gegensatz zu den bisherigen Verfahren ausdrücklich um große Mengen multivariater Daten erfassbar zu machen und ist damit, wiederum im Gegensatz zu den bisher beschriebenen Methoden,

eindeutig der explorativen Datenanalyse zuzuordnen. In der Literatur finden sich leicht unterschiedliche Beschreibungen des Verfahrens. Grundsätzlich kann aber gesagt werden, dass bei diesem Verfahren die Darstellung wie in einer Tabellenkalkulation geschieht. Dabei werden die Daten aber teilweise grafisch dargestellt und teilweise ausgeblendet. Die Daten eines Teils einer Tabellenzeile werden grafisch dargestellt. Die angrenzenden Zeilen und Spalten werden fortschreitend bis unter Umständen zur Unleserlichkeit kleiner. Auf diese Art und Weise können zahllose Datensätze mit mehreren Dutzend Variablen visualisiert werden. Diese Methode unterscheidet sich auch dadurch von den bisherigen, dass sie nur in einer interaktiven Umgebung, also zum Beispiel an einem PC, einsetzbar ist. Ähnliche Verfahren sind *Bifocal view* und *Fisheye view*, bei denen ebenfalls die kontextrelevanten bzw. benachbarten Informationen angezeigt werden, die weiter von den aktuell ausgewählten Elementen jedoch nicht.

Diese beispielhafte Liste von Methoden könnte problemlos um ein vielfaches ausgedehnt werden. Wichtig ist jedoch, dass das in Kapitel 2.1 genannte oder ein ähnliches Verfahren angewendet wird um eine für den jeweiligen Zweck passende Visualisierungsmethode zu finden. Die dabei zu berücksichtigenden Kriterien lauten unter anderem: Anzahl der Dimensionen der Daten, Zweck der Visualisierung (explorative Analyse,...), Art bzw. Möglichkeiten zur Interaktion, Vorkenntnisse der Rezipienten etc. Eine anschauliche Sammlung von Visualisierungsmethoden findet sich im Internet unter dem Namen *Periodic Table of Visualization Methods*[1], zahlreiche weitere Beispiele sollten mit den Suchworten Infovis oder Visual Analytics zu finden sein.

Gütekriterien für Visualisierungsdesign

Tufte hat sich schon 1983 der schwierigen Aufgabe gewidmet Gütekriterien für Informationsvisualisierungen aufzustellen. Die vier wesentlichen und kontextrelevanten Kriterien sollen hier zusammengefasst dargestellt werden.

Graphical Excellence Nach Tufte zeichnet sich eine exzellente Darstellung von Information durch Klarheit, Präzision und Effizienz aus. Visualisierungen sollten ihm zu Folge also unter anderem:

- Nur die benötigten Daten abbilden
- Den Rezipienten zum Nachdenken über die Daten und nicht die Darstellungsmethode anregen
- Verzerrte Darstellungen bzw. Fehlwahrnehmungen vermeiden
- Eine hohe Informationsdichte aufweisen

[1] http://www.visual-literacy.org/periodic_table/periodic_table.html,zuletzt abgerufen 26.03.2011

- Den Vergleich verschiedener Datensätze ermöglichen
- Verschiedene Detaillierungsgrade wahrnehmbar machen
- Zweckorientiert sein, also zum Beispiel explizit eine deskriptive oder explorative Wahrnehmung ermöglichen

[Tuf83, S. 13 ff]

Mazza bezeichnet in diesem Kontext eine gute Informationsvisualisierung als eine interessante Präsentation von Daten, die sich durch Substanz und Design auszeichnen.[Maz09, S. 13] Während die hier aufgestellten Unterkriterien kaum messbar und dazu noch unter Umständen konfliktär sind (z.B. größte Anzahl Daten auf kleinstmöglichem Raum), sind die weiteren Gütekriterien konkreter.

Graphical Integrity Dieses Kriterium, das Tufte aufstellt, fasst Mazza sehr prägnant zusammen: Sie beschreibt die wissenschaftliche Güte der Visualisierung, die keine falsche oder verzerrte Interpretation zulassen darf. So soll zum Beispiel bei der Darstellung numerischer Daten die Darstellung immer proportional erfolgen. Auch soll die Anzahl der Dimensionen der Darstellung auf keinen Fall die Anzahl der Dimensionen der Daten überschreiten. Der Richtigkeitsanspruch gilt auch für die Legenden. Mazza stellt die These auf, dass viele Informationsvisualisierungen mit eher künstlerischem Anspruch erstellt werden, ohne das der Ersteller fundierte statistische Kenntnisse hat, so dass eher kreative Artefakte als klare, eindeutige Visualisierungen entstehen. Tufte stellt für dieses Kriterium eine mittlerweile berühmte Kennzahl auf, den so genannten „Lie Factor“[Tuf83, S. 57], der ein Maß für die Falschdarstellung von Daten in einer grafischen Darstellung benennt. Er wird durch das Teilen der Effektgröße in den Daten (z.B. die Steigung) durch das Teilen der Effektgröße in der Darstellung berechnet. Diese Kennzahl soll natürlich in einer idealen Visualisierung gleich Null sein, es finden sich aber zum Beispiel in den Medien sehr viele Beispiele für Visualisierungen, bei denen dieser Lie Factor sehr hohe Werte annimmt. Dies kann einerseits auf eine manipulative Datendarstellung hindeuten oder aus Unkenntnis in der Anwendung von Darstellungsmethoden sein.[Tuf83, S. 57 f.]

Maximized Data-Ink Ratio Die so genannte *Data-Ink Ratio* bezeichnet eine hypothetische Kennzahl, die das Verhältnis von dargestellter Information zur für die Visualisierung benutzten Tinte bezeichnet. Da weder die Menge der visualisierten Daten in diesem Kontext sinnvoll quantifiziert werden kann, und auch die benutzte Tinte, bzw. die genutzten Pixel keine Aussagekraft haben, ist das streben nach der Maximierung dieser Kennzahl als gestalterisches Ideal zu betrachten. Grundsätzlich sollen Visualisierungen minimalistisch gestaltet sein. Alle unnötigen Elemente, also solche die keine Information vermitteln, wie Rahmen, Hintergründe, 3D-Effekte, sollen vermieden werden. Dieses

Kriterium lässt sich teilweise auch aus dem ersten Kriterium, der Graphical Excellence ableiten.[Tuf83, S.93 ff] oder [Tuf83, S.14]

Aesthetics Das Kriterium der Ästhetik ist sicherlich das am schwierigsten zu beschreibende. Tufte stellt jedoch wieder einige Leitlinien bzw. Ideale auf, an Hand deren sich dieses Kriterium erschließt und die im praktischen Gestaltungsprozess berücksichtigt werden können. Die wesentlichen lauten nach Tufte:

- Adäquates Format (auch im Sinne der gewählten Methodik)
- Multimodalität der Visualisierung (Wörter, Zahlen, Grafik)
- Narrative Qualität
- Qualitativ hochwertig und präzise hergestellt

[Tuf83, S. 177]

Am Beispiel der gewünschten narrativen Qualität einer Visualisierung wird deutlich, dass künstlerischer Anspruch und wissenschaftliche bzw. wahrnehmungspsychologische Qualität einer Visualisierung verschwimmen. Tufte stellte diese Kriterien immerhin in einer Zeit auf, in der Computer noch kaum verbreitet waren und die Möglichkeiten der digitalen Datenverarbeitung noch sehr eingeschränkt waren. Dennoch haben die aufgestellten Kriterien noch Gültigkeit. Der Aspekt der Interaktivität ist jedoch der gestiegenen Verbreitung und Leistungsfähigkeit von modernen Computern zu verdanken und soll separat im nächsten Kapitel bearbeitet werden.

Interaktion mit Informationsvisualisierungen

Dadurch, dass aktuelle Computer große Mengen an Daten sehr schnell verarbeiten können und auch grafische Darstellungen sehr schnell erzeugen können, ist es heute möglich Visualisierungen zur explorativen Datenanalyse interaktiv zu gestalten. Dies bedeutet einen enormen Fortschritt im wissenschaftlichen Feld der Visual Analytics. Im Folgenden werden einige grundlegende Interaktionsmöglichkeiten kurz dargestellt.

Durch *Interaktive Filterung*, auch Interactive Filtering bzw. Information-Hiding der dargestellten Daten können für die Untersuchung nicht relevante Daten ausgeblendet werden. Dieses Verfahren bedeutet im Kontext der Interaktivität dass die dargestellte Datenmenge ohne größeren Aufwand wie zum Beispiel das komplette Neuerstellen der Visualisierung verändert werden kann.[Maz09, S. 117 f.]

Navigationsfunktionen, wie zum Beispiel das Vergrößern der Abbildung, also Zoomen und damit verbunden auch das Verschieben des sichtbaren Bereiches können genutzt werden um dem Rezipienten erst einen Überblick zu ermöglichen und dann Bereiche um bestimmte Werte genauer betrachten zu können.[Noc07, S. 14]

Overview & Detail ist eine mit der vorherigen Methode verwandte Option. Während der Nutzer sich bestimmte Bereiche vergrößert anschauen kann, wird parallel dazu eine Überblicksansicht angezeigt, auf der erkennbar ist welchen Bereich der Gesamtheit der Daten bzw. der Abbildung der Nutzer sich grade anschaut. In der Regel kann mit Hilfe dieser Überblicksanzeige navigiert werden, der Detailansichtbereich also verschoben werden.[Nor05, S. 21ff]

Focus & Context bezeichnet Methoden, bei denen, ähnlich wie bei Overview & Detail eine bestimmte Region der Abbildung vergrößert dargestellt wird. Allerdings werden hier zusätzlich detailliertere Informationen angezeigt. Für die Bereiche die nicht im Fokus der Betrachtung liegen werden verkleinert oder aggregiert angezeigt. Die oben beschriebene Visualisierungsmethode Table Lens macht sich diese Interaktionsmethode zu nutze.[Maz09, S. 110]

Als *Brushing* bezeichnet man ein Verfahren, bei alle Werte bzw. Eigenschaften die einer bestimmten Gruppe angehören hervorgehoben werden. Dies geschieht zum Beispiel durch eine unterschiedliche Färbung. Dadurch können bestimmte Teilmengen von Daten zum Beispiel auf Korrelationen hin untersucht werden, ohne dass die anderen Daten ausgeblendet werden müssen. Durch die Verwendung mehrerer Farben kann auch eine Vergleichbarkeit von mehreren Teildatenmengen hergestellt werden. Weitergehend kann Linking eingesetzt werden um die ausgewählte Datenmenge in mehreren Visualisierungen hervorzuheben.[Noc07, S. 14]

2.5 Frameworks

Dieses Kapitel des Abschlussberichtes beschäftigt sich mit möglichen Nutzbaren Visual Analytic Frameworks. Dabei erfolgt eine Diskussion über die Anforderungen, die ein brauchbares Visual Analytics Framework benötigt und damit erste Kriterien für die in die nähere Auswahl kommenden Frameworks liefert. Im Anschluss daran werden verschiedene Visual Analytics Frameworks vorgestellt für die Programmiersprache C# da sich für diese am Anfang der Entwicklung in der Projektgruppe entschieden wurde.

2.5.1 Anforderungen an Visual Analytics Frameworks

Aufgrund der Voraussetzung das BMW Datalogs betrachtet und analysiert werden, ist eine der Anforderungen die Verarbeitung großer Datenmengen. Weiterhin soll das später genutzte Framework eine übersichtliche Darstellung der Daten ermöglichen. Diese Darstellung soll sowohl Graphen, Diagramme als auch Bäume als Darstellungstypen bereitstellen beziehungsweise eine einfache Integration dieser ermöglichen. Zusätzlich ist eine verhältnismäßig leichte Navigation innerhalb der Datenmengen und eine dynamische Neustrukturierung gewünscht. Bestenfalls ist der Übergang zwischen Übersichtsdarstellung und einer detaillierten Darstellung bestimmter Datenareale stufenlos durchzuführen.

2.5.2 Vorstellung der Frameworks

Im Bereich der Visual Analytics existiert eine Vielzahl an *Frameworks*, jedoch kommen die meisten für eine nähere Betrachtung nicht in Frage. Häufig werden diese entweder nicht mehr gepflegt oder sind nur schwer erweiterbar, durch schlechte oder fehlende Dokumentation der Komponenten. In diesem Teil der Arbeit werden die Frameworks GraphViz, Piccolo2D und der Prototyp Microsoft Vedea vorgestellt, die in C# implementiert wurden oder aber sich in C# einbinden lassen. Zusätzlich soll am Anfang dieses Abschnitts die Windows Presentation Foundation vorgestellt werden mit dem die Visual Analytics Frameworks verknüpft werden sollten können um einfacher die Multitouchoperationen die das Windows Surface SDK zur Verfügung stellt verwenden zu können. Zudem bietet die Windows Presentation Foundation auch schon von sich aus einige Möglichkeiten zur Visualisierung von Daten die genutzt werden könnten.

Windows Presentation Foundation

Die Windows Presentation Foundation (WPF) ist Teil des .NET-Frameworks 3.0 und höher und bietet den Vorteil einer Hardwarebeschleunigten Darstellung, da sofern eine DirectX kompatible Grafikkarte vorhanden ist Direct3D zum zeichnen der Inhalte genutzt wird. Das entwerfen von Anwendungen ist sowohl mit Microsoft Visual Studio 2010, 2008 ,2005 mit Erweiterung und SharpDevelop möglich. Diese genannten Entwicklungsumgebungen beinhalten für die Erstellung WYSIWYG(What You See Is What You Get)-Designer und XAML-Editoren. Bei XAML handelt es sich um eine an XML angelehnte Struktur zur Beschreibung der WPF-GUIs. Zum Designen von Oberflächen gibt es das Werkzeug *Microsoft Expression*, das zur Gestaltung von WPF-GUIs verwendet werden kann. Zusätzlich existiert *Microsoft Expression Blend*, bei dem es sich um ein professionelles Interface-Designwerkzeug handelt, welches zur

Gestaltung von modernen Benutzeroberflächen für Desktopanwendungen verwendet wird.[KJ10]

GraphViz

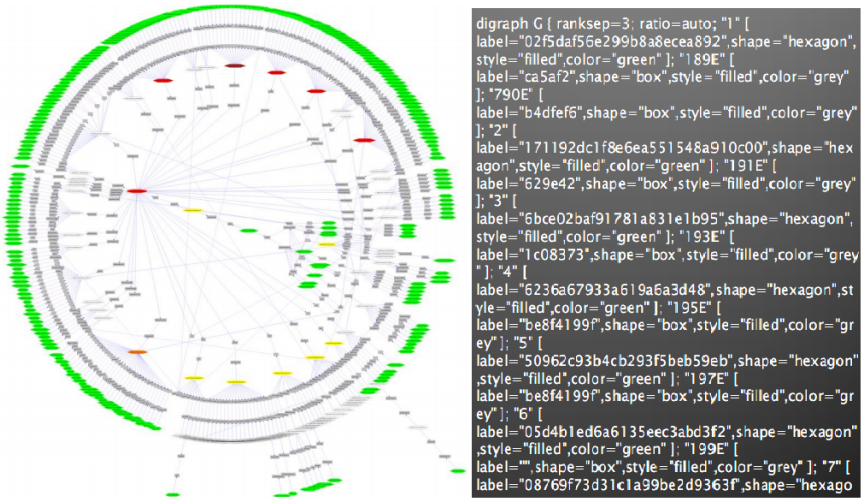


Abbildung 2.11: graphviz: Beispielgraph mit Ausschnitt der beschreibenden Datei

Das *GraphViz* (http://www.graphviz.org/) Framework wurde von ATT und den Bell-Labs als plattformübergreifendes Framework zur Visualisierung von Objekten und deren Beziehungen untereinander entwickelt. Formal ausgedrückt visualisiert GraphViz gerichtete und ungerichtete Graphen. Alle zur Erzeugung einer Grafik benötigten Anweisungen werden dabei aus einer Textdatei geladen. Diese Beschreibung beinhaltet dabei die Beschreibung der Knoten und Kanten des Graphen. Die Positionen der Knoten, sowie die Krümmungen der Kanten werden aus dieser Beschreibung automatisch berechnet und dabei so optimiert, dass die Struktur des Graphen erkennbar ist. Zur Beschreibung des darzustellenden Graphen wird die Auszeichnungssprache DOT verwendet, die sich syntaktisch an die Programmiersprache C anlehnt. GraphViz bietet bei Bedarf weitere Möglichkeiten zur Veränderung des Layouts, der Form und Farbgestaltung des Graphen. Oft genügt allein die Strukturdefinition des Graphen zur Erzeugung einer für den Menschen übersichtlichen Ausgabe. Daher können nicht nur Menschen, sondern auch automatische Prozesse GraphViz zur Erstellung von Visualisierungen nutzen (z.B. Doxygen). Auch können an vorhandenen Graphen sehr schnell Veränderungen über die DOT-Datei vorgenommen werden, was mit einem Standard-Grafikprogramm nicht ohne weiteres möglich ist. Jedoch werden im Falle von GraphViz nur Bilder aus einer DOT-Datei generiert, weswegen große Datenmengen nicht in Echtzeit exploriert werden können. In Abbildung 2.11 befindet sich ein mit

GraphViz erstellter *Hyperbolic Tree*, mit dazugehörigem Ausschnitt der beschreibenden Textdatei [Ell03].

Piccolo2D

Das Piccolo2D(http://www.piccolo2d.org/) Framework ist ein Toolkit, dass im Allgemeinen das Zeichnen von 2D Grafiken unterstützt. Eine große Besonderheit des Piccolo2D Frameworks ist das integrierte Zoomable User Interface(ZUI). Ein ZUI ist eine Art von Interface, das eine große Menge an Informationen darstellen kann, in dem es den Nutzer stufenlos reinzoomen lässt um detailliertere Informationen zu bekommen und rauszoomen lässt um einen Überblick zu erhalten. Dieses wurde über ein *Scene Graph*-Model realisiert, welches an die 3D- Programmierung angelehnt ist und nur Teile des Graphen zeichnet, die auch gerade auf dem Bildschirm Sichtbar sind. Das Piccolo2D Framework nimmt dem Programmierer, dabei insbesondere die Arbeit mit LowLevel-Details ab wie zum Beispiel das optimierte Zeichnen der Visualisierung. Es existieren drei Versionen des Frameworks zum einen Piccolo2D.NET, PocketPiccolo2D.NET und Piccolo2D.Java. In Abbildung 2.12 sieht man die Darstellung verschiedener mit Piccolo2D realisierter Graphen. [BGM04][Der05]

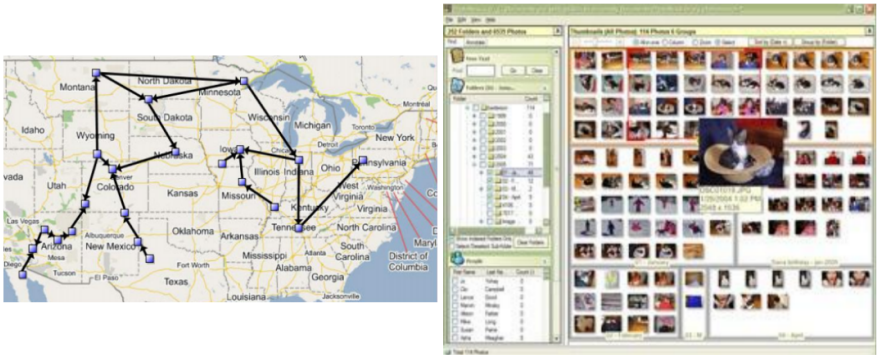


Abbildung 2.12: Piccolo2D-Graphen: Links im Bild Graph der Reiseroute auf einer USA-Karte darstellt; rechts im Bild Treemap zur Darstellung ähnlicher Bilder

(Microsoft Vedea)

Zu den Frameworks, die in diesem Abschnitt noch eine gesonderte Betrachtung finden sollen -auch wenn es noch nicht verfügbar ist- gehört Microsoft Vedea, das unter dem Namen Microsoft Visualization Language oder Vedea Projekt bekannt ist. Dabei handelt es sich um den Prototypen einer neuen experimentellen Sprache zum Erstellen von interaktiven *Infographics*. Diese wird vom Computational Science Laboratory(Microsoft

Research; http://research.microsoft.com/en-us/groups/science/) entwickelt, insbesondere für Personen ohne Programmierkenntnisse und soll eine Erleichterung bei der Erstellung interaktiver Visualisierungen bieten. Dabei muss vom Entwickler keine Betrachtung von Zeichnungs- und Visualisierungs-APIs erfolgen. Da von einer zeitlich nahen Veröffentlichung als Community Technical Preview ausgegangen wird und ebenfalls eine API zur Verfügung gestellt werden soll, wurde dieses Projekt ebenfalls als interessantes Visual Analytics Framework angesehen. Insbesondere da die Verwendung zweier Microsoft Produkte sowohl für die Gestenerkennung als auch für die visuelle Analyse eine stabilere Grundlage bieten würde. [UNK10]

2.6 Zusammenfassung

Im vorangegangenen Kapitel wurden die Grundlagen der visuellen Analyse beschrieben. Bei dieser Methode werden Daten visuell dargestellt, um den Menschen stärker in den Analyseablauf mit einzubeziehen. Dies ist der große Vorteil im Gegensatz zu algorithmischen Verfahren, da der Mensch leicht Strukturen oder Zusammenhänge in den Visualisierungen wahrnehmen kann. Desweitern wurde der „Visual-Analytics-Prozess“ vorgestellt. Dieser Prozess analysiert die Daten mit automatischen und visuellen Verfahren. Dies führt zu einer kontinuierlichen Weiterentwicklung und Überprüfung der Ergebnisse. Irreführende Ergebnisse können somit in einem frühen Stadium entdeckt werden.

In dem Unterkapitel „Nutzen“ werden die hauptsächlich die Vorteile durch Hinzunahme des Menschen in den Visual Analytics Prozess dargestellt. Die wichtigsten Kriterien dafür sind die Flexibilität, die Kreativität sowie das Allgemeinwissen des Menschen mit den Vorzügen der heutigen Computertechnologie zu vereinen.

Im Folgenden wurden die visuellen Analysetechniken in die drei Sparten „zu visualisierende Datentypen“, „Visualisierungstechniken“ und „Interaktionstechniken“ klassifiziert und Beispiele für diese einzelnen Teilbereiche vorgestellt. Im Kapitel „Visualisierungsdesign“ wurden Erkenntnisse der Informationsvisualisierung für Anwendungen der visuellen Analyse dargestellt.

Zum guter Letzt wurden Visual Analytics Frameworks für die Programmiersprache C# vorgestellt. Außerdem wurden die Anforderungen, die ein brauchbares Visual Analytics Framework benötigt, diskutiert und somit die ersten Kriterien für die Auswahl der infrage kommenden Frameworks geliefert.

Kapitel 3

Surface Computing

Dieses Kapitel gibt einen kleinen Überblick über das Thema *Surface Computing*.
Surface Computing umfasst das Arbeiten mit dem Computer und die Interaktion über eine Oberfläche (engl. *Surface*), in den meisten Fällen den Bildschirm. Zunächst wird im Abschnitt *State of the Art* ein kurzer Überblick darüber gegeben, welche Geräte zur Zeit im Bereich des Surface Computing auf dem Markt erhältlich sind. Diese sind im Besonderen in die Kategorien *SmartPhones* und *All-in-One-PCs* unterteilt. Anschließend erfolgt eine kleine Einführung in das Thema *Multi-Touch* und was man darunter versteht. Im Abschnitt *Visuelle Analyse auf Surface Computern* wird kurz darauf eingegangen, wie man durch den Einsatz von Surface Computing die visuelle Analyse von Daten unterstützen kann. Darauf folgend werden im Teil *Frameworks* einige Umgebungen für die Entwicklung von Anwendungen für das Surface Computing gegenübergestellt, wobei hier ein besonderes Augenmerk auf Umgebungen mit C#-Unterstützung gelegt wird.

3.1 State of the Art

Aktuell befinden sich immer mehr Geräte auf dem Markt, die sich durch Berührung mit einem oder mehrerer Finger bedienen lassen. Dabei stechen insbesondere Mobiltelefone, die *SmartPhones*, und *All-In-One-PCs* hervor.

SmartPhones

SmartPhones sind Mobiltelefone der neusten Generation, die weit mehr können, als einfach nur Anrufe zu tätigen und SMS verarbeiten. Es handelt sich dabei eher um vollwertige Computer im Hosentaschenformat. Bei vielen dieser Geräte wird weitestgehend auf die Eingabe mit Knöpfen verzichtet und sie lassen sich durch einen berührungsempfindlichen Bildschirm steuern.
Mittlerweile ist eine große Fülle an verschiedenen Modellen von diversen Anbietern

auf dem Markt. Diese unterscheiden sich meist nur durch kleine Details und durch die in ihnen eingesetzten Technologien. Während beispielsweise Apple für das *iPhone 4* das in Eigenentwicklung entstandene Betriebssystem *iOS 4* verwendet [App11], wird beim *Samsung Galaxy S I 9000* das Betriebssystem *Android 2.1* verwendet [Sam11]. Als weitere Alternativen stehen neben anderen noch *Blackberry OS*, *Symbian S60* und *Windows Mobil/Phone* zu Verfügung, welche in verschiedenen Modellen Anwendung finden [Gmb10b].
Je nach verwendetem Betriebssystem unterscheiden sich die Anwendungsmöglichkeiten. So gibt es für jedes der genannten Systeme verschiedene Programme, die aus dem Internet heruntergeladen werden können. Die wohl bekanntesten Vertreter stellen hier die sogenannten *Apps* aus dem *AppStore* von Apple dar. Der Begriff *App* ist eigentlich die Kurzform für das englische Wort *application* (Anwendung), wird jedoch oft als Synonym für SmartPhone-Programme verwendet.
Ein weiteres Unterscheidungsmerkmal der einzelnen SmartPhones - neben den physikalischen Eigenschaften, wie Gewicht und Größe, und beispielsweise der Auflösung der Bildschirme - ist die darin verwendete Technologie. So gibt es mehrere verschiedene Möglichkeiten die bei SmartPhones übliche Berührungserkennung zu realisieren. Die beiden Technologien mit dem größten Marktanteil sind hierbei die kapazitiven Sensoren und die resistiven Sensoren (s. Abb.3.1).
Bei ersteren wird im Bildschirm ein Spannungsfeld aufgebaut. Durch die Berührung mit

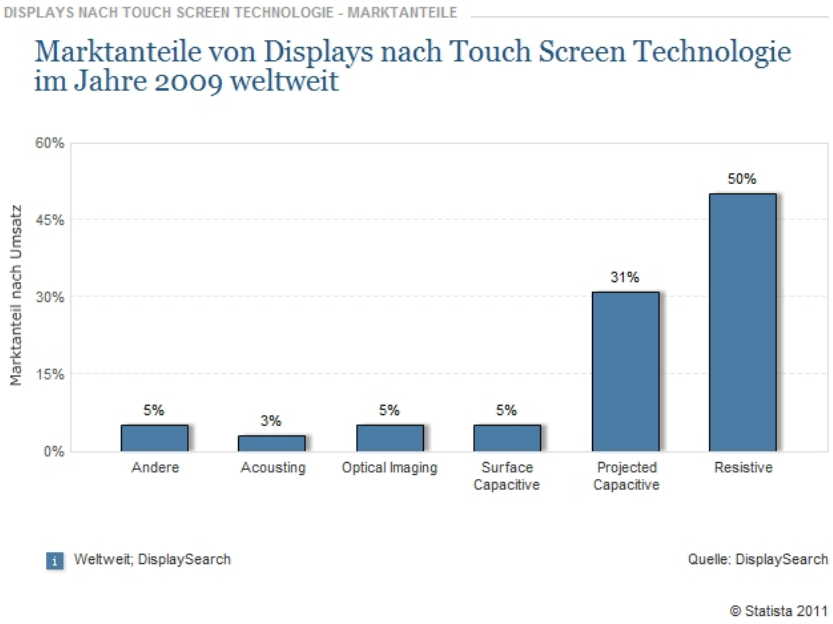


Abbildung 3.1: Marktanteile Multi-Touch-Technologien 2009 nach [Gmb10a]

einem Finger kommt es im Display zu einem Spannungsabfall. Dieser Spannungsabfall

wird dann als Eingabe interpretiert (s. Abb. 3.2).
Bei resistiven Sensoren liegen auf dem Display zwei Schichten übereinander, die sich jedoch nicht berühren. Beide Schichten stehen unter Spannung. Der Druck mit einem Finger oder einen beliebigen Gegenstand auf das Display bewirkt, dass sich beide Schichten berühren. Der Berührungspunkt wird dann als Eingabe interpretiert (s. Abb. 3.3) [NZ07].

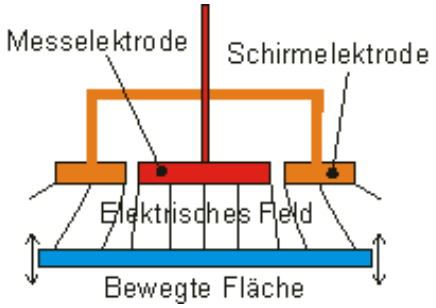


Abbildung 3.2: Funktionsweise kapazitativer Sensor nach [Fac11]

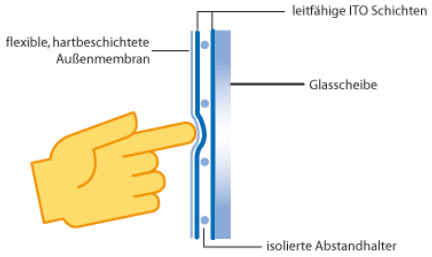


Abbildung 3.3: Funktionsweise resistiver Sensor nach [EK08]

All-In-One-PCs

Ein All-In-One-PC vereint in sich den klassischen Heimcomputer mit seinen Peripheriegeräten in einem Gerät. Durch die Bedienung über den berührungsempfindlichen Bildschirm kann auf externe Eingabegeräte, wie Maus und Tastatur, verzichtet werden. Die Hardware eines PC kann durch die flache Bauweise von Flachbildschirmen im selben Gehäuse untergebracht werden. So entsteht ein Gerät von den Ausmaßen eines kleinen Fernsehers mit den Funktionalitäten eines Heimcomputers. Diese Geräte werden oft auch als Fernseher mit Internetzugang beworben und weniger als Ersatz für einen „echten“ PC. So werden diese Geräte oftmals auch mit einer Fernbedienung ausgestattet, so dass sie bequem vom Sofa aus bedient werden können, wie beispielsweise der *Medion The Touch x9613* [AG11b].
Eine alternative Bauweise hierzu ist das Integrieren der Geräte in einem Tisch, so dass die Tischoberfläche als Bildschirm und Eingabegerät dient. Durch das größerer Platzangebot in einem Tisch kann hier eine weitere Bildschirmtechnologie eingesetzt werden, die theoretisch beliebig viele Eingabepunkte gleichzeitig erkennen und verarbeiten kann. Hierbei wird die Tischoberfläche von unten mit einer Kamera abgefilmt, die mit einem Infrarotfilter ausgestattet ist. Die Tischfläche selbst besteht dabei aus zwei Schichten. Die oberste Schicht ist eine dünne Silikonschicht und darunter befindet sich eine mittels Infrarot-LEDs angestrahlte Schicht. Das Infrarotlicht verbleibt in dieser Schicht, bis die Silikonschicht eingedrückt wird. Das Eindrücken bewirkt, dass das Infrarotlicht am

Berührungspunkt gestreut wird. Diese Streuung wird von der Kamera aufgezeichnet und durch den angeschlossenen Rechner in Eingabesignale umgewandelt [CFK+10, S.32].

3.2 Multitouch

Bei der Bedienung eines Gerätes wird generell in zwei Kategorien unterteilt. Diese Kategorien lassen sich als *sequentiell* und *parallel* bezeichnen. Bei der sequentiellen Bedienung erfolgen die Eingaben nacheinander, bilden also eine Sequenz. Das Tippen eines Textes, der nur aus Kleinbuchstaben besteht, kann als sequentielle Eingabe verstanden werden. Demgegenüber steht die parallele Eingabe, bei der mehrere Eingaben gleichzeitig erfolgen können. Als Beispiel kann hier das Tippen eines deutschen Textes dienen, bei dem zum Beispiel für die Verwendungen von Großbuchstaben die Benutzung der Umschalttaste hilfreich ist. Ohne die Möglichkeit, mehrere Tasten gleichzeitig zu verwenden, müsste der Umweg über die Feststelltaste erfolgen.
Diese grundlegende Unterscheidung zwischen sequentieller und paralleler Bedienung gibt es auch bei *Touch Screens*. Die Entwicklung von (Multi)Touch-Computern begann bereits in den 1960er Jahren. Die meisten der ersten Vertreter dieser Produkte konnten nur einen einzigen Berührungspunkt gleichzeitig erkennen. Sogar das erste Apple iPhone unterstützte nur eingeschränkt gleichzeitige Eingaben [Bux11]. Heutige Geräte unterstützen fast alle die parallele Bedienung durch mehrere Finger. Dies ist nicht nur praktisch, um beispielsweise mit zwei Fingern in einer Grafik zoomen zu können, sondern auch notwendig, wenn auf einem Surface Computer ein Text mit einer virtuellen Tastatur eingegeben werden soll.

3.3 Visuelle Analyse auf Surface Computern

Ein typischer Computer-Arbeitsplatz ist meist als Einzelarbeitsplatz optimiert. Die Bildschirmauflösung ist meist sehr gut, aber die Bildschirmgröße und der eingeschränkte Betrachtungswinkel beeinträchtigt die Teamarbeit erheblich. Darum ist das Arbeiten auf einer größeren interaktiven Benutzeroberfläche ein wichtiger Schritt, um das kollaborative Arbeiten zu optimieren.
Dabei können mehrere Visualisierungen von mehreren Personen gleichzeitig bearbeitet und im Team neue Erkenntnisse aus diesen gewonnen werden. Die optimale Ausrichtung einer interaktiven Benutzeroberfläche zum kollaborativen Arbeiten, ist die horizontale Position. Die tischförmige Gestalt bietet viel Platz, um von allen Seiten an dem *Multi-Touch-Tisch* zu arbeiten. Durch das Ausführen der visuellen Datenexploration auf einer interaktiven Benutzeroberflächen, können die Analysten direkt mit dem Diagramm interagieren. Dies führt zu einer dynamischeren Art der Analyse und gibt dem Benutzer

das Gefühl, eine bessere Kontrolle über seine Tätigkeit zu haben. Der Analyst hat zur Exploration verschiedene Operationen zur Verfügung wie zum Beispiel das Zoomen und das Verfeinern, sowie ein neues Diagramm erstellen zu können und viele weitere.

3.4 Frameworks

Frameworks bieten einem Anwendungsentwickler verschiedene Hilfsmittel und Möglichkeiten, seine Anwendungen effizienter und einfacher zu gestalten. Im Bereich der Entwicklung von Anwendungen für das Surface Computing und Multi-Touch sind einige Frameworks erhältlich. Hier wird nun eine Auswahl davon gegenüber gestellt. Zu Beginn des Projektes haben wir uns dazu entschieden, als Programmiersprache C# zu verwenden. Daher werden hier nun nur Frameworks für C# betrachtet.

WPF - Windows Presentation Foundation

Die *WPF* ist Teil von .NET 3 und höher und bietet seit Version 4 ein Framework zur Entwicklung von Multi-Touch-Anwendungen auf Basis von C#. Microsoft setzt bei WPF auf eine zweigeteilte Entwicklung. So können die Frontend-Elemente, wie beispielsweise die Benutzeroberfläche, über XAML erstellt werden. Bei XAML handelt es sich um einer speziell für Anwendungen konzipierten Weiterentwicklung von XML. Das Verhalten des Frontends kann über den dahinter liegenden Code in beispielsweise C# festgelegt werden [Mos11]. Neben C# kann bei WPF auch Visual Basic, C++, und F# verwendet werden. WPF ist speziell zur Entwicklung von Windows-Anwendungen konzipiert und bietet von sich aus keine Unterstützung für Linux-Derivate und andere Betriebssysteme. Außerdem wird für die Benutzung *Visual Studio 2010* vorausgesetzt. WPF unterstützt nativ das Verarbeiten von Windows 7 Touch-Ereignissen und liefert die wichtigsten Basiselemente zur Anwendungsentwicklung mit. Die grafische Ausgabe, insbesondere von 3D-Objekten, basiert auf *Direct3D*.

Silverlight

Bei *Microsoft Silverlight* handelt es sich um ein WPF-Leichtgewicht zur Entwicklung von Web- Anwendungen mit Visual Studio 2010. Seit Version 4 unterstützt Silverlight auch die Verarbeitung von Touch-Ereignissen und kann auch zur Entwicklung von nicht-browserbasierten Anwendungen benutzt werden. Da Silverlight in erster Linie ein Framework zur Erstellung von browserbasierten Anwendungen ist, ist Microsoft hier auf Kompatibilität mit vielen Browsern und Betriebssystemen bedacht. So unterstützt Silverlight neben dem *Internet Explorer* auch die aktuellsten Versionen von *FireFox*,

Safari und *Google Chrome*. Die mitgelieferten Basiselemente sind insbesondere für den Gebrauch in Browsern zugeschnitten [Mic11b].

Surface Toolkit for Windows Touch Beta

Das *Surface Toolkit for Windows Touch Beta* stellt eine Erweiterung für WPF dar und bietet einige zusätzliche Optionen für die Entwicklung von Touch-Anwendungen mit WPF. Wie bei WPF wird hier für die Entwicklung Visual Studio 2010 und .NET 4.0 benötigt. Das Surface Toolkit ist insbesondere für die Entwicklung von Anwendungen für *Microsoft Surface* konzipiert und unterstützt ausschließlich Derivate von Windows 7 [Mic11a]. Mit der Einführung von *Surface 2.0* soll das Surface Toolkit for Windows Touch Beta durch das *Surface 2.0 SDK* abgelöst werden [Mic11c].

3.5 Zusammenfassung

Die Geschichte des Surface Computing reicht in die 1960er Jahre zurück und erlebte erst im letzten Jahrzehnt den Marktdurchbruch. Insbesondere in diesen zehn Jahren kam es zu einem regelrechten Boom, so dass es nun eine große Reihe von entsprechenden Geräten auf dem Markt gibt, die für jedermann erschwinglich sind. Entsprechend viele Anwendungsgebiete und Entwicklungsumgebungen gibt es. Hier wurden insbesondere Umgebungen aus dem Hause Microsoft vorgestellt, die vorwiegend für die Entwicklung von Windows-Anwendungen konzipiert sind. Es gibt jedoch auch entsprechende Gegenstücke für beispielsweise *Java*, welche weitestgehend plattformunabhängig sind. Die visuelle Analyse konnte durch den Einsatz von Surface Computern einen gewaltigen Schritt nach vorne machen und wird sich wohl in Zukunft noch weiter entwickeln, so dass es dem Menschen immer besser möglich sein wird, seine intuitive Auffassungsgabe und Mustererkennung durch den Einsatz solcher Systeme zu unterstützen. So können immer größere Datenmengen in kürzerer Zeit semi-automatisch erfasst und verarbeitet werden. Hierbei wird insbesondere auf Synergieeffekte gesetzt, die entstehen, wenn die menschliche Intuition mit computergestützter Datenverarbeitung kombiniert wird.

Kapitel 4

User Centered Design

In diesem Kapitel wird beschrieben, wie unter Berücksichtigung von Humanfaktoren, mit geeigneten Evaluationsmethoden und nach den Prinzipien des Usability Engineerings das System auf den Nutzer zugeschnitten werden kann.

Im ersten Abschnitt werden dabei die Fähigkeiten und Grenzen der Leistungsfähigkeit von Menschen in einem Mensch-Maschine-System beschrieben. Diese werden häufig als „Human Factors“ oder „Humanfaktoren“ bezeichnet. Im zweiten Abschnitt wird das User Centered Design beschrieben und konkreter erläutert welche Herausforderungen bei der Zusammenarbeit mit einem großen Unternehmen auftreten. Im dritten Abschnitt werden anschließend verschiedene Evaluationsmethoden präsentiert.

4.1 Human Factors

In klassischen Vorgehensmodellen werden Systeme so entwickelt, dass sie mit gegebener Hardware möglichst effektiv und effizient funktionieren. Um mit so einem System zu arbeiten, muss die Bedienung erlernt werden. Da der Umfang der zur Verfügung gestellten Funktionen stetig zunimmt, wird das bloße Erlernen der Funktionen schwieriger. Werden Humanfaktoren berücksichtigt, ist die Bedienung natürlicher und dadurch leichter zu erlernen. Außerdem wird die Wahrscheinlichkeit einer Fehlbedienung reduziert. Ein weiterer Vorteil ist, dass die technische Umgebung mithilfe der Humanfaktoren optimal an den Menschen angepasst werden kann, was zu einer erhöhten Gebrauchstauglichkeit führt. Sind die Fähigkeiten und Schwächen von Mensch und Maschine bekannt, können auch die Aufgaben besser verteilt werden. Während beispielsweise komplexe Rechenoperationen vom Computer übernommen werden, wird eine Mustererkennung durch den Menschen effektiver durchgeführt.

Einen allgemeinen Überblick über Human Factors liefert zum Beispiel das Buch *Human-Computer Interaction* von Dix et al., dessen Inhalte zum Teil auch im Internet verfügbar

sind [DFAB04]. Einige Ergebnisse wie ein Nutzerinterface mit Gesten auf einem Surface Computer realisiert werden können, präsentieren Wobbrock et al. [WMW09b].

Human Factors sind nicht nur physische Fähigkeiten, sondern auch psychische, kognitive und soziale. Im Folgenden werden nur die visuellen Faktoren vorgestellt, da diese eine besonders wichtige Rolle bei der visuellen Analyse auf einem Surface Tisch einnehmen. Trotzdem werden alle Faktoren bei der Entwicklung nach den Prinzipien des Usability Engineerings berücksichtigt und evaluiert.

4.1.1 Visuelle Faktoren

In diesem Abschnitt wird zunächst grob skizziert, wie die Wahrnehmung funktioniert. Anschließend wird gezeigt, wie einfach die Wahrnehmung getäuscht werden kann. Um daraufhin auf die *Gestaltpsychologie* einzugehen, die ein Ansatz ist, um die Wahrnehmung in Regeln zusammenzufassen.

Das Sehen

Wenn Licht durch unser Auge auf die Netzhaut trifft, wird ein chemischer Prozess in Gang gesetzt, der dazu führt, dass im Gehirn elektrische Signale ankommen, die dann interpretiert werden [DFAB04, GK03]. Die Anzahl der Bilder, die pro Sekunde wahrgenommen werden können und welche Farben gesehen werden können unterliegt dadurch den folgenden Regeln.

Wahrnehmung von Einzelbildern Der chemische Prozess, der das Sehen ermöglicht, kann nur mit einer begrenzten Geschwindigkeit reagieren. Diese sogenannte *Flimmerfusionsfrequenz* liegt bei Menschen etwa bei 10-70 Hz [HF64, Wik10a], was bedeutet, dass spätestens ab etwa 70 Bildern pro Sekunde davon ausgegangen werden kann, dass das Auge keine Einzelbilder mehr wahrnehmen kann. Wie viele Bilder das Gehirn pro Sekunde tatsächlich verarbeiten kann, ist unbekannt. Allerdings werden bei Animationen Objekte im Bild mit einer Geschwindigkeit von 24 bis 30 Bildern pro Sekunde abgespielt. Dabei wird ausgenutzt, dass Bildwechsel in dieser Geschwindigkeit bereits als *flüssig* wahrgenommen werden [BDR00]. Daher kann vorausgesetzt werden, dass das Gehirn normalerweise höchstens 30 Bilder pro Sekunde als Einzelbild erkennen kann. Wird die Flimmerfusionsfrequenz zusätzlich in Betracht gezogen, kann davon ausgegangen werden, dass höchstens zehn Bilder pro Sekunde einzeln wahrgenommen werden können, während erst ab 30 Bildern pro Sekunde flüssige Übergänge simuliert werden können. In dem Bereich dazwischen können zwar meist keine Einzelbilder ausgemacht werden, aber die Bewegung wirkt flackernd und unnatürlich.

Wahrnehmung von Farben Nicht alle Farben werden mit der gleichen *Auflösung* wahrgenommen. Vielmehr sind zum Beispiel mehr Rottöne voneinander unterscheidbar, als Blautöne. In Abbildung 4.1 ist der sichtbare Farbbereich unabhängig von der Helligkeit dargestellt. Zu Problemen bei der Unterscheidung verschiedener Farben kommt es bei *Farbenfehlsichtigkeit*. Etwa acht bis neun Prozent der Männer und ein Prozent der Frauen sind farbenfehlsichtig, was bedeutet, dass sie nicht alle Farben sehen können. Etwa 99% von ihnen können nicht zwischen rot und grün unterscheiden, was je nach Zielgruppe beim Design berücksichtigt werden muss.

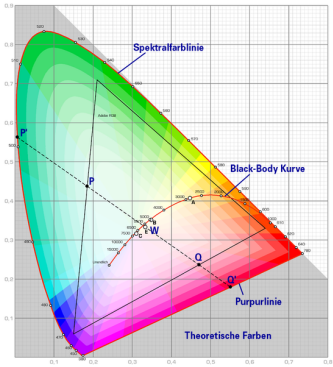


Abbildung 4.1: CIE Normfarbtafel

Optische Täuschungen

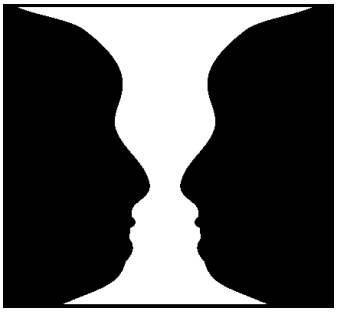


Abbildung 4.2: *Kippbild* als Beispiel zur Figur-Grund Relation

In Abbildung 4.2 ist ein *Kippbild* dargestellt, bei dem entweder zwei Gesichter im Vordergrund zu sehen sind oder eine Vase. Solche „Fehler“ in der Wahrnehmung entstehen häufig nicht im Auge, sondern im Gehirn.

Bach und Poloschek unterstützen die These, dass optische Täuschungen fest im Gehirn verankert sind und uns in Alltagssituationen Vorteile geben [BP06]. Damit Täuschungen,

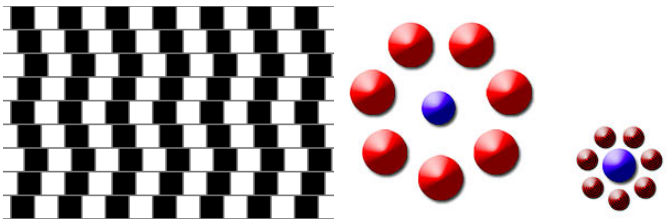


Abbildung 4.3: Weitere Illusionen

wie in Abbildung 4.3 gezeigt, nicht bei der Visualisierung von Analysedaten auftreten, sondern diese im besten Fall ausgenutzt werden können, ist es wichtig sie zu verstehen.

Gestaltpsychologie

Es gibt mehrere Möglichkeiten zu beschreiben, wie das Gehirn automatisch ein Bild so ändert, dass es einen „Sinn“ ergibt. Eine Möglichkeit dazu ist die Gestalttheorie, wie sie am Anfang des letzten Jahrhunderts von Carl Stumpf, Max Werheimer, Wolfgang Köhler und Kurt Koffka formuliert wurde [Wik10b, Gus96, e-t07, Hic09]. Diese Gesetze müssen beachtet werden, um falsche Interpretationen zu vermeiden, können aber auch ausgenutzt werden, um Beziehungen zu visualisieren, wie es von Hicks [Hic09] gezeigt wird. Im Folgenden wird gezeigt, wie die Gestaltgesetze definiert sind.

Gesetz der Prägnanz Das Gesetz der *Prägnanz* oder der *guten Gestalt* beschreibt, dass jede Figur so wahrgenommen wird, als würde sie aus möglichst einfachen Strukturen bestehen. In Abbildung 4.4 werden daher links fünf Kreise gesehen, die von einem weißen Rechteck überdeckt werden und rechts ein Kreis, das von einem schwarzen Rechteck überdeckt wird (oder umgekehrt), obwohl dort nur sechs Objekte vorhanden sind.

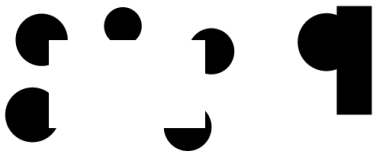


Abbildung 4.4: Beispiel für das Gesetz der Prägnanz

Gesetz der Nähe Nach dem Gesetz der *Nähe*, werden Objekte als zusammengehörig interpretiert, die nahe beieinander liegen. In Abbildung 4.5 werden also statt vieler Kreise links eine große Gruppe von Kreisen und rechts drei Gruppen von Kreisen gesehen. Das kann ausgenutzt werden, indem ähnliche Werte nah beieinander gesetzt werden.

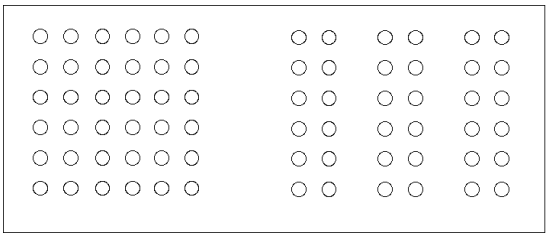


Abbildung 4.5: Beispiel für das Gesetz der Nähe

Gesetz der Ähnlichkeit Nach dem Gesetz der *Ähnlichkeit* oder *gleicher Merkmale*, werden Objekte zu Gruppen zusammengefasst, die ähnliche Eigenschaften teilen. In Abbildung 4.6 werden daher die leeren und die ausgefüllten Kreise jeweils zu Gruppen zusammengefasst. Wenn ähnliche Werte also ähnliche Farben erhalten, können beispielsweise Ausreißer und zusammengehörige Daten schnell erkannt werden.

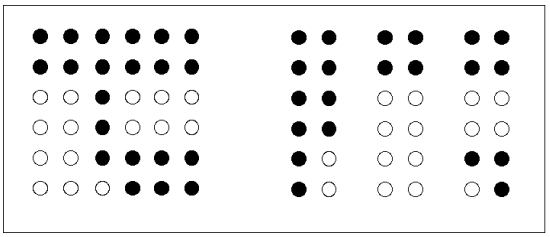


Abbildung 4.6: Beispiel für das Gesetz der Ähnlichkeit

Gesetz der gemeinsamen Region Laut dem Gesetz der *gemeinsamen Region* werden Objekten die gemeinsam in einem abgegrenzten Gebiet liegen als Gruppe empfunden. In Abbildung 4.7 werden daher die Kreise in den Rechtecken zusammengefasst. Dies kann als weitere Möglichkeit genutzt werden Daten zu gruppieren.

Gesetz der Kontinuität Nach dem Gesetz der *Kontinuität* wird erwartet, dass Muster sich Fortsetzen. Daher werden in Abbildung 4.8 zwei gepunktete Linie erkannt, statt einzelne Punkte. Nach diesem Gesetz könnten Werte nach einem bestimmten System angeordnet werden, sodass Lücken oder andere Abweichungen von der Erwartung schnell erkannt werden.

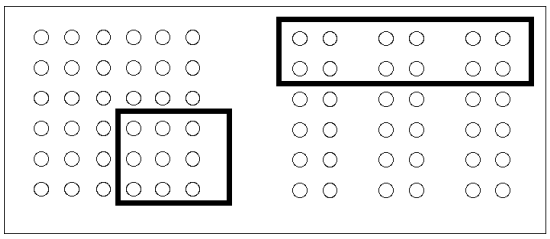


Abbildung 4.7: Beispiel für das Gesetz der gemeinsamen Region

Abbildung 4.8: Beispiel für das Gesetz der Kontinuität

Gesetz der Symmetrie Laut dem Gesetz der *Symmetrie* werden Objekte als verbunden empfunden, die sich entlang einer Achse aufeinander spigeln lassen. In Abbildung 4.9 sind daher die beiden Kurven links als stärker zusammenhängend empfunden, als das bei den beiden rechten Kurven der Fall ist, obwohl die sich ähnlicher sind. Wenn also beispielsweise zwei Kurven verglichen werden sollen, ist es sinnvoll eine der Kurven zu spiegeln, damit die (fehlende) Symmetrie schnell auffällt.

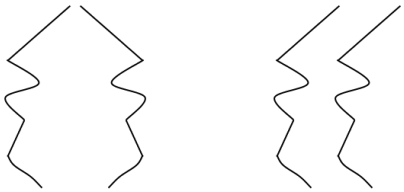


Abbildung 4.9: Beispiel für das Gesetz der Symmetrie

Gesetz der Geschlossenheit Das Gesetz der *Geschlossenheit* sagt aus, dass geschlossene Objekte eher als Einheit gesehen werden, als andere. In Abbildung 4.10 sind daher statt vielen senkrechten und zwei waagerechten Linien mehrere Gruppen von senkrechten Linien und ein Rechteck zu sehen. Für eine Visualisierung könnte das zum Beispiel genutzt werden, um die Punkte in einer Figur zu verschmelzen, statt sie nach dem Gesetz der Region zu gruppieren.

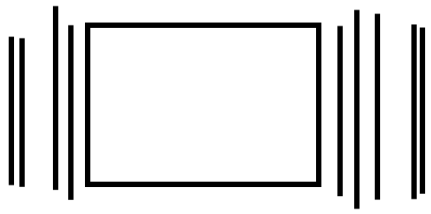


Abbildung 4.10: Beispiel für das Gesetz der Geschlossenheit

Gesetz der verbundenen Elemente Das Gesetz der *verbundenen Elemente* sagt aus, dass Elemente gruppiert werden die verbunden sind. Das wird in Abbildung 4.11 dargestellt. Trotz ihrer Nähe, sind die linke und die rechte Gruppe klar voneinander getrennt. Dies kann benutzt werden, um Elemente stärker aneinander zu binden, ohne sie zu verändern oder zu verschieben.

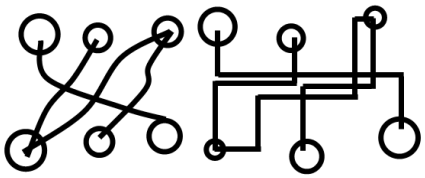


Abbildung 4.11: Beispiel für das Gesetz der fortgesetzt durchgehenden Linie und der verbundenen Elemente

Gesetz der fortgesetzt durchgehenden Linie Nach dem Gesetz der *duchgehenden Linie*, nehmen wir immer an, das Linien fortgesetzt werden und nicht abknicken, wenn sie sich mit einer anderen Linie kreuzen. In Abbildung 4.11 wird gezeigt, dass die Zuordnung der Kreise links einfacher ist, als rechts, da das Gesetz ausgenutzt wurde. Das kommt bei der Visualisierung zum Tragen, wenn es darum geht, wie sich Linien überschneiden dürfen.

Gesetz der gemeinsamen Bewegung Das Gesetz der *gemeinsamen Bewegung* oder des *gemeinsamen Schicksals* besagt, dass Objekte gruppiert werden, die sich gemeinsam Bewegen. Würden sich beispielsweise in Abbildung 4.5 zwei der Punkte bewegen, würden diese sich damit vom Rest abheben. Bei Visualisierungen, die zum Beispiel eine Zeitachse haben, kann nach diesem Gesetz einfach erkannt werden, welche Objekte sich über einen Zeitraum in die gleiche Richtung bewegen und somit zusammen gehören.

4.2 Usability Engineering

In diesem Abschnitt geht es darum, wie Visualisierungen entworfen werden können. Zunächst wird dabei das *User Centered Design* vorgestellt. Daraufhin werden Gestaltungsgrundsätze gemäß DIN EN ISO 9241-110 vorgestellt und schließlich wird auf Empfehlungen eingegangen, die Sedlmair et al. gegeben haben, um die Anforderungen mit großen Firmen zu ermitteln [SIBB10].

4.2.1 User Centered Design

Wenn das System möglichst optimal auf den Benutzer ausgerichtet sein soll, muss dieser von Anfang an so oft wie möglich mit einbezogen werden. Das Ziel dieses sogenannten *User Centered Designs* ist es die Nutzbarkeit zu maximieren, indem der Nutzer in jeder Phase der Entwicklung einbezogen wird. Dazu muss zunächst der Nutzungskontext analysiert werden. Als nächster Schritt werden die Anforderungen definiert, daraufhin wird ein erster Prototyp erstellt, der dann evaluiert wird. Die Ergebnisse dieser Evaluation werden anschließend ausgewertet und fließen wieder in die Anforderungen ein. Diese Schritte werden so lange wiederholt, bis die Gebrauchstauglichkeit, oder *Usability*, den Ansprüchen der Nutzer gerecht werden. Eine Standardisierung für dieses Vorgehen bietet die ISO-13407 [Sta11], die in Abbildung 4.12 dargestellt ist.

Anforderungen

Teile der Anforderungen sind meist schon von außen gegeben (zum Beispiel technische Grenzen). Ist noch nicht ganz klar definiert, was das System leisten soll, bietet es sich an mit den Benutzern Ideen zu sammeln. Zum Beispiel mithilfe von Interviews oder Mind-Mapping.

Benutzeranalyse

Während der Nutzeranalyse wird festgehalten, welche Gruppen von Nutzern es gibt. Diese werden nach ihren Charakteristiken wie zum Beispiel Wissensstand, Erfahrung und Gewohnheiten gruppiert. So kann dann herausgefunden werden, ob es innerhalb der Zielgruppe Untergruppen gibt, die einen anderen Entwurf benötigen. Dazu eignet sich aber zum Beispiel Interviews, oder die Befragung von Experten, die über die tatsächlichen, eventuell unbewussten Gewohnheiten der Zielgruppe besser Bescheid wissen.

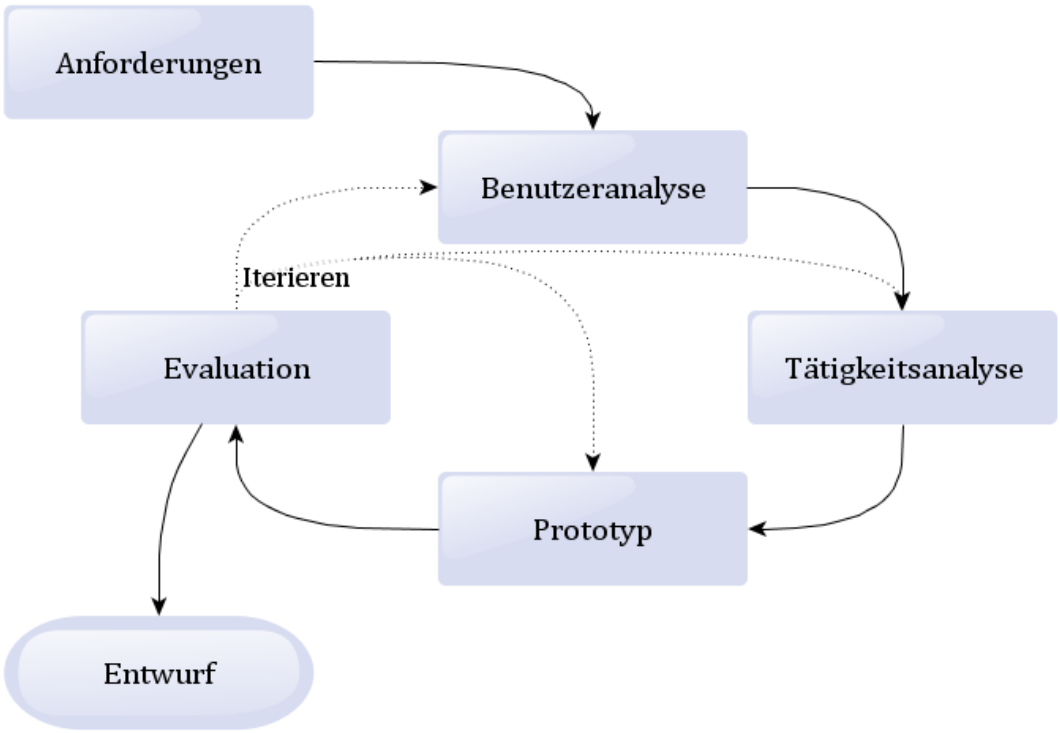


Abbildung 4.12: User Centered Design nach ISO 9241-210

Tätigkeitsanalyse

Die Tätigkeitsanalyse gehört im Prinzip zur Nutzeranalyse. Dabei wird beispielsweise herausgefunden, wie Benutzer bestimmte Funktionen benutzen. Hierbei hilft zum Beispiel eine *Contextual Inquiry*, bei der der Nutzer über einen längeren Zeitraum beobachtet wird.

Prototyp

Aus den bisherigen Erkenntnissen wird mindestens ein Prototyp entwickelt, der alles enthält, was im folgenden Schritt evaluiert werden soll. Dabei reicht unter Umständen bereits ein einfacher Papier-Prototyp, beispielsweise um ein bestimmtes Layout gegen ein anderes zu testen.

Evaluation

Bei der Evaluation wird normalerweise entweder eine neue Lösungsidee gegen eine alte oder mehrere alternative Lösungsideen gegeneinander getestet. Dabei können die Prototypen zum Beispiel von den Nutzern mit einem Usability-Test, wie dem System Usability Scale (SUS), bewertet werden. Die Ergebnisse fließen dann wieder in die Anforderungen mit ein.

Entwurf

Stellt sich nach einer Evaluation heraus, dass die Lösung ausreichend gebrauchstauglich ist und allen Anforderungen genügt, wird sie für den Entwurf des Systems übernommen. Ansonsten wird eine neue Iteration bei der Benutzeranalyse, der Tätigkeitsanalyse, oder beim Prototypen gestartet.

4.2.2 Gestaltungsgrundsätze gemäß DIN EN ISO 9241-110

Eine gute Anwendung sollte derart gestaltet sein, dass die Anwender diese intuitiv bedienen und erlernen können. Daher ist es notwendig einige Gestaltungsgrundsätze während des Designprozesses zu beachten.

Zu diesem Zweck beschreibt der Teil 110 der DIN EN ISO 9241 folgende sieben Grundsätze für die Gestaltung und Bewertung der Anwendung. [The10, S.233]

Aufgabenangemessenheit „*Ein interaktives System ist aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h., wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie.*“ [Sch10a]

Dies bedeutet, dass nur die Informationen angezeigt werden sollen, die in einem Zusammenhang mit der zu bearbeitenden Aufgabe stehen. Daher sollen alle überflüssigen Informationen nicht dargestellt werden, denn diese lenken von der eigentlichen Information ab. Zudem ist es sinnvoll in entsprechenden Eingabefeldern bereits Standardwerte vorzugeben, da diese das Bearbeiten der Aufgabe erleichtern.

Selbstbeschreibungsfähigkeit „*Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle im Dialog er sich befindet, welche Handlungen unternommen werden können und wie diese ausgeführt werden können.*“ [Sch10f]

Um ein System selbstbeschreibungsfähig zu gestalten sollte beispielsweise auf übliche und einheitliche Symbole gesetzt werden. Des Weiteren sollte für den Anwender jederzeit erkennbar sein, in welchem Bearbeitungsschritt dieser sich befindet und welche weiteren Schritte durchgeführt werden müssen, um die Arbeitsaufgabe erfolgreich abzuschließen.

Lernförderlichkeit „*Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen der Nutzung des interaktiven Systems unterstützt und anleitet.*“ [Sch10e]

Unter Lernförderlichkeit daher verstanden, dass der Anwender jederzeit bei der Benutzung des Systems unterstützt werden soll. Dies kann beispielsweise dadurch gegeben werden, dass dem Anwender die Möglichkeit geboten wird die Dialogschritten erneut auszuführen oder Rückmeldungen zu Zwischen- und Endergebnissen erfolgen.

Steuerbarkeit „*Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.*“ [Sch10g]

Demnach wird die Steuerbarkeit dadurch bestimmt, ob dem Anwender in jedem Eingabefeld die Möglichkeit gegeben wird, seine Eingabe rückgängig zu machen. Aber auch, ob ihm die Wahl des Ein- und Ausgabemediums gegeben wird. Folglich soll der Anwender zu jeder Zeit die volle Kontrolle über die einzelnen Schritte besitzen.

Erwartungskonformität „*Ein Dialog ist erwartungskonform, wenn er den aus dem Nutzungskontext heraus vorhersehbaren Benutzerbelangen sowie allgemein anerkannten Konventionen entspricht.*“ [Sch10b]

Demzufolge ist mit der Erwartungskonformität gemeint, dass der Benutzer beispielsweise eine einheitliche Verwendung von Strukturen und Anordnungen der Eingabefelder, Menüs und Buttons vorfindet, wie diese bei anderen Anwendungen ebenso dargestellt sind. Aber auch an Stellen, wo der Benutzer Rückmeldungen erwartet, diese entsprechend ausgegeben werden, zum Beispiel eine Bestätigung oder Hinweis zu einen längeren Ladevorgang.

Individualisierbarkeit „*Ein Dialog ist individualisierbar, wenn Benutzer die Mensch-System-Interaktion und die Darstellung von Informationen ändern können, um diese an ihre individuellen Fähigkeiten und Bedürfnisse anzupassen.*“ [Sch10d]

Jedem Benutzer sollte daher die Möglichkeit geboten werden, das System auf seine eigenen Bedürfnisse und Wünsche umstellen zu können. So sollten zum Beispiel Menüleisten an und abschaltbar sein oder an einer anderen Stelle angezeigt werden können. Aber auch die Möglichkeit einzelne Elemente einer Menüleiste neu zu ordnen oder hinzuzufügen, sollte gegeben werden.

Fehlertoleranz „*Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.*“ [Sch10c]

Das System sollte fehlerhafte Benutzereingaben erkennen und anzeigen sowie bei der Fehlerbehebung und Vermeidung unterstützen. Dabei sollten Fehleingaben nicht zu Abbrüchen des Systems führen und die Schritte zur Fehlerbehebung minimal sein. Unterstützend ist hierbei sinnvoll, dass die Daten vor der Verarbeitung auf Korrektheit und Gültigkeit geprüft werden.

4.2.3 Vorgehen bei großen Unternehmen

Wie das Vorgehen bei dem Entwurf von Visualisierungen mit einem User Centered Design aussehen kann, zeigen Wassink et al. [WKD+08]. In diesem Abschnitt sollen nun aber die Herausforderungen bei der Zusammenarbeit mit einem großen Unternehmen gezeigt werden, die nicht bereits in den Prinzipien des Usability Engineering formuliert wurden.

Die Vorlage für den Umgang mit diesen Besonderheiten liefern Sedlmair et al. [SIBB10]. Sie hielten sich drei Jahre bei einem großen Unternehmen auf, um dort Visualisierungen zu Analysezwecken zu entwickeln. In ihrer Arbeit wird beschrieben, welche Erfahrungen sie dabei gemacht haben und was bei Studien mit großen Unternehmen beachtet werden sollte.

„Mach dich mit der aktuellen Software Umgebung vertraut“

Neben einer Analyse des aktuellen Stands der Technik, wird empfohlen abzuwägen, ob das zu entwickelnde System ein neuer Teil der Tool-Chain werden soll oder ob das System so entwickelt werden kann, dass ein bereits vorhandenes Glied der Tool-Chain ersetzt werden kann.

„Überwinde technische Hürden bei der Datenintegration“

In der Industrie ist Zeit ein sehr wichtiger Faktor. Daher ist es wichtiger einen Datentyp sicher und schnell zu unterstützen, als zu versuchen möglichst viele Datentypen zu unterstützen, die zuerst in einen internen Datentyp konvertiert werden müssen, was mehr Zeit kostet.

„Wähle deine Studienumgebung mit Bedacht“

Ebenso, wie es besser ist, sich auf wenige Datentypen zu beschränken, ist es einfacher eine gute Lösung für eine Untergruppe der Zielgruppe zu entwickeln, als zu versuchen eine Lösung zu finden, die für alle Gruppen passt. Bei der Wahl der Untergruppe sollte auch beachtet werden, welche organisatorischen und politischen Probleme sich ergeben könnten.

„Die magische Eine-Stunde-Grenze“

Es ist signifikant einfacher Teilnehmer für maximal eine Stunde zu finden als für längere Zeiträume. Eine Studie oder Evaluation muss also so aufgebaut sein, dass sie innerhalb einer Stunde abgeschlossen werden kann. Je kürzer, desto besser.

„Überzeuge deine Zielgruppe“

Die Zielgruppe wird wesentlich interessierter und motivierter, wenn ihre aktuell real existierenden Probleme gelöst werden. Wenn möglich könnten real existierende Probleme durch Interviews mit der Zielgruppe herausgefunden werden. Mindestens eine Lösung dieser Probleme sollte eine hohe Priorität haben, auch wenn es das Projekt nicht unbedingt wesentlich weiter bringt.

„Begeistere mit Usability und Ästhetik“

Es ist wichtig, dass die „Schönheit“ und Usability nicht unterschätzt wird, da sie einen großen Einfluss darauf haben, ob ein System als „gut“, oder „schlecht“ empfunden wird.

„Lerne von den Experten“

Experten werden meist kein großes Interesse an neuen Lösungen haben, da sie eigene entwickelt haben. Trotzdem ist es wichtig herauszufinden, warum ihre Lösungen gut sind, um das in der eigenen Lösung zu berücksichtigen. Wenn es machbar ist, sollte für das Projekt zumindest ein Experte befragt werden.

„Versuche eine Lizenz zu bekommen, führe Studien aber auf jeden Fall durch“

Es ist einfacher die Ergebnisse einer Studie auszuwerten, wenn Audio- und Video-Aufzeichnungen vorliegen. Bevor eine Studie mit Aufzeichnungen durchgeführt wird, muss zuerst mit der Firma geklärt werden, ob aufgezeichnet werden darf. Außerdem dürfen nur Personen an der Studie teilnehmen, die mit der Aufzeichnung einverstanden sind. Gibt es Probleme damit, ist eine Alternative mit mehreren Beobachtern zu arbeiten, die sich jeweils Notizen machen und diese direkt im Anschluss zusammenfassen.

„Bleibe stets in engem Kontakt“

Um die teilweise sehr komplexen Bereiche zu überblicken sollte ein enger Kontakt zur Zielgruppe bestehen. Dadurch kann zum Beispiel schnell bei Problemen nachgefragt werden.

„Die magische Maßeinheit: Geld“

Wissenschaftliche Ergebnisse sind zwar interessant, aber um die Stakeholder davon zu überzeugen das Projekt zu unterstützen, müssen monetarisierbare Einheiten, wie *Entscheidungen pro Stunde*, *Gefundene Fehler pro Tag*, oder ähnliche Zeit, bzw. Geld sparende Maßnahmen gezeigt werden. Für dieses studentische Projekt ist es nicht von hoher Priorität Stakeholder zu überzeugen. Trotzdem ist es von Vorteil, wenn im Ergebnis gezeigt werden kann, dass zum Beispiel die *gefundenen Fehler pro Tag* durch das neue System erhöht werden konnten.

„Rechne hohen Skill bei bekannten Techniken mit ein“

Einige erfahrene Teilnehmer könnten schnell lernen, wenn ein altes System gegen ein neues getestet wird. Dies kommt bei Studien zum Tragen und muss entsprechend beim Design der Studie berücksichtigt werden.

„Kläre vorher wie die Ergebnisse veröffentlicht werden dürfen“

Es ist wichtig, dass vor Beginn der Studie geklärt wird in welcher Form Ergebnisse präsentiert werden dürfen. Dabei muss auch berücksichtigt werden, welche Bilder verwendet werden dürfen, ob Ergebnisse nur anonymisiert veröffentlicht werden darf und ob Veröffentlichungen zuerst durch ein internes Review gehen müssen.

4.3 Evaluationsmethoden

Wie bereits in den Kapiteln zuvor beschrieben, ist die Durchführung von Evaluationen ein geeignetes Mittel, um eine gute Gebrauchstauglichkeit einer Software zu erreichen oder diese gegebenenfalls zu verbessern.

Im Folgenden werden daher fünf verschiedene Methoden aufgezeigt.

4.3.1 Heuristische Evaluation

Die heuristische Evaluation ist eine der bekanntesten Methoden, um die gebrauchstaugliche Gestaltung eines Systems zu überprüfen. Dabei gehört diese Methode, wie auch die Methode des Cognitive Walkthrough, zu den expertenorientierten Methoden. [Nac08, S. 27ff]

Entwickelt wurde die Methode der heuristischen Evaluation von Jakob Nielsen sowie Rolf Molich und wird im Idealfall von drei bis fünf Usability-Experten durchgeführt. [Nie] Während der Evaluationstudie überprüfen mehrere Experten die Benutzerschnittstelle auf ihre Übereinstimmung mit den zehn von Nielsen entworfenen Prinzipien, welche auch Heuristiken genannt werden. [Nac08, S. 27ff]

Diese zehn Heuristiken sind wie folgt beschrieben:

- Visibility of system status,
- Match between system and the real world,

- User control and freedom,
- Consitency and standards,
- Error prevention,
- Recognition rather than recall,
- Flexibility and efficiency of use,
- Aesthetic and minimalist design,
- Help user recognize, diagnose, and recover from errors,
- Help and documentation. [Nie05]

Während der Durchführung überprüfen die Experten mehrmals das System auf die Übereinstimmung mit den zehn Heuristiken. Dabei ist zu beachten, dass jeder Experte die Studie alleine durchführt. Erst nachdem alle Evaluationsstudien durchgeführt sind, ist es den Experten erlaubt untereinander zu kommunizieren und ihre Ergebnisse zusammenzufassen. Dies ist wichtig, damit die einzelnen Experten bei deren Durchführung nicht beeinflusst werden und somit unabhängige Ergebnisse gewährleistet werden [Nie].

4.3.2 Cognitive Walkthrough

Die Evaluationsmethode des *Cognitive Walkthrough* gehört zu den expertenorientierten Methoden und wurde von Polson und Lewis entwickelt. [Kus09, S. 18ff]

Im Vordergrund stehen bei dieser Evaluationsmethode vor allem die mentalen Prozesse, die bei einem Benutzer synchron zu den Arbeitsschritten ablaufen und weniger die grafische Benutzeroberfläche. Daher ist es wichtig, dass sich der Experte gut in die Rolle des späteren Benutzers versetzen kann, denn nur so kann eine erfolgreiche Evaluation gewährleistet werden. Während der Evaluation durchläuft der Experte vorgegebene Handlungsabläufe, die zuvor analysiert wurden. Zudem wird davon ausgegangen, dass der Benutzer kein produktspezifisches Vorwissen besitzt und somit das explorierende Lernen in dieser Evaluationsstudie im Vordergrund steht. Dabei überprüft der Experte, ob es dem Benutzer möglich ist, den optimalen Lösungsweg für ein Problem zu finden. [Nac08, S. 27] Weiterhin wird festgestellt und überprüft, welches Vorwissen der Benutzer benötigt, damit das System fehlerfrei bedient werden kann und nicht an Interaktionsproblemen zu scheitert. Um dies herauszufinden ist von dem zu evaluierenden System zumindest ein Papierprototyp erforderlich, wodurch die Benutzungsschnittstelle nachempfunden werden kann [Kus09, S. 19f].

Das Ziel des Cognitive Walkthrough besteht somit darin, dem Designer und Entwickler des Systems die Verbesserungspotentiale für die Gestaltung der Benutzungsschnittstelle aufzuzeigen [Nac08, S. 28].

4.3.3 Thinking-Aloud

Bei der Methode des *Thinking-Aloud* handelt es sich nicht um eine expertenorientierte Evaluationsmethode. Hier können, im Gegensatz zum heuristischen Verfahren so wie zum Cognitive Walkthrough, bereits Evaluationen mit „normalen“ Probanden durchgeführt werden.

Dabei ist die Methode des Thinking-Aloud mittlerweile ein Standardverfahren, das sich als gute Evaluationsmethode etabliert hat. Am geeignetsten ist diese Methode dafür, kognitive Probleme der Benutzer beim Erlernen sowie in der Bedienung des Systems aufzudecken. Während der Studie sind die Probanden dazu aufgefordert ihre Gedanken kontinuierlich laut auszusprechen. Dadurch wird das Ziel verfolgt, die mentalen Eindrücke des Probanden nachvollziehen zu können und in Folge dessen Schwierigkeiten und Probleme, bspw. in der Bedienung, aufzudecken. Daher kann durch diese Methode erfahren werden, wie bestimmte Komponenten des Systems durch die Benutzer interpretiert werden [Wag04, S. 160f].

4.3.4 System Usability Scale

Neben der zuvor beschriebenen Methode bietet sich außerdem die Durchführung einer Evaluation mit Hilfe von Fragebögen an. Fragebögen besitzen im Gegensatz zu anderen Evaluationsmethoden den Vorteil, dass diese relativ schnell ausgewertet werden können und daher schneller ein Ergebnis zur Verfügung steht. So kann die Auswertung unter anderem automatisch geschehen. Jedoch besteht bei Fragebögen die Gefahr, dass die Fragen, bspw. aufgrund schlechter Formulierung, falsch verstanden werden und das Ergebnis dadurch nachteilig beeinflusst wird [Züh04, S. 131].

Ein Fragebogen, welcher gute Ergebnisse liefert, ist der *System Usability Scale* (SUS). Der System Usabilty Scale ist ein standardisierter Fragebogen, der aus zehn vordefinierten Fragen besteht. Die darin enthaltenden Fragen sind jeweils abwechselnd positiv und negativ, in Betracht zum bewertenden System, gestellt [Bro].

Der Ablauf dieses Fragenbogens ist wie folgt: Zuerst füllt der Proband den Fragebogen, nach seinen persönlichen Eindrücken über das System, aus. Im Rahmen dessen besteht die Auswahl der Antwortmöglichkeiten von stark zustimmend bist stark ablehnend. Nachdem alle Fragen ausgefüllt worden sind, wird der Fragebogen nach einem bestimmten Schema ausgewertet [Bro].

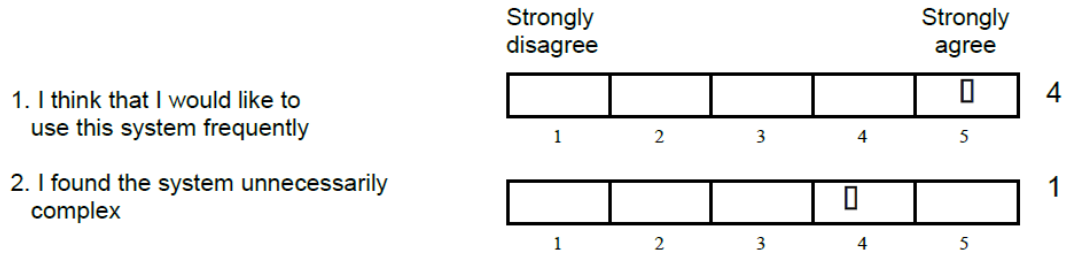


Abbildung 4.13: Zwei Beispielfragen des SUS [Bro]

Die dargestellte Abbildung 4.13 verdeutlicht den Wechsel der jeweils positiv und negativ gestellten Fragen. Innerhalb der ersten Frage wird danach gefragt, ob der Proband das System öfters benutzen würde. Sofern der Proband hierbei das Feld 5 ankreuzt, wird die erste Frage mit einer Gewichtung von vier angegeben. Wäre das Feld 1 von dem Probanden angekreuzt worden, hätte die erste Frage eine Gewichtung von null erhalten. Dies ist zugleich das Vorgehen bei der Auswertung einer positiv gestellten Frage. Denn eine starke Zustimmung der ersten Frage, stellt das Gesamtsystem besser dar, als eine starke Ablehnung. Anders ist es hingegen bei der zweiten Frage. Bereits anhand der Formulierung (Ich fand das System unnötig komplex) ist ersichtlich, dass es sich hierbei um eine negativ gestellte Frage handelt. Bei einer Bewertung der Frage mit Feld 5, würde der Proband der Aussage vollkommen zustimmen und das System im Gegensatz zur ersten Frage schlecht darstellen. Damit bei der Endauswertung des Fragebogens fehlerfreie Ergebnisse erzielt werden, muss diese Frage bei einer starken Zustimmung eine andere Bewertungsstruktur erhalten, als bei der ersten Frage. Hierbei ist daher darauf zu achten, dass negativ gestellte Aussagen, anders als positiv gestellte Fragen, bei der Bewertung mit Feld 5, eine Gewichtung von null erhalten. Weiterhin erhält eine starke Ablehnung, der negativ gestellten Frage, eine Gewichtung von vier [Bro].

Nachdem alle zehn Fragen von dem Probanden bewertet wurden, findet die Endauswertung des Fragebogens statt. Dazu werden zuerst alle Gewichtungen, der einzelnen Fragen, nach dem zuvor beschriebenen Schema ermittelt. Diese zehn ermittelten Gewichtungen werden summiert und können dadurch eine Gesamtgewichtung im Wertebereich von 0 bis 40 ergeben. Um nun die Endbewertung des System Usability Scale zu erhalten, wird der Wert der Gesamtgewichtung mit 2,5 multipliziert und ergibt damit einen Wert von 0 bis 100. Dabei gilt, je höher dieser Wert ist, umso besser wurde das System bewertet. Des Weiteren kann mittels dieses Wertes das System mit anderen Bewertungen verglichen werden, und so bspw. herausgefunden werden, ob die Usability des Systems verbessert wurde [Bro].

4.3.5 Eye-Tracking

Im Gegensatz zu den vorangegangenen Evaluationsmethoden, setzt das *Eye-Tracking* Verfahren auf technische Hilfsmittel, die die Augen- bzw. Blickbewegungen des Probanden aufzeichnen und auswerten. [Hei10, S. 117] Ein solches technisches Hilfsmittel ist, wie in der folgenden Abbildung 4.14 dargestellt, beispielsweise eine Brille, die verschiedene Kameras nutzt, um den Blickverlauf des Probanden aufzunehmen. Dazu nimmt eine Kamera das Blickfeld des Probanden auf und zwei andere Kameras die Bewegungen der Pupillen. Dadurch kann berechnet werden, welche Bereiche eines Systems der Proband aktuell betrachtet [tea10].

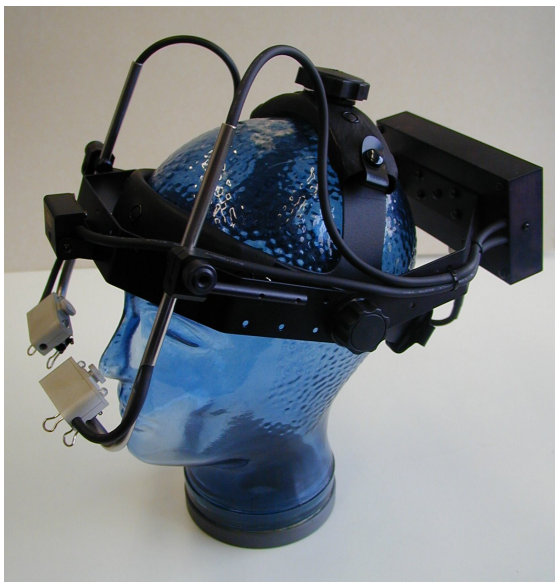


Abbildung 4.14: Ein Beispiel für eine Eye-Tracking Kamera [tea10]

Bei moderneren Verfahren des Eye-Tracking, müssen die Probanden keine Kamera auf dem Kopf tragen. Hier sind die Kameras direkt in den Rahmen der Monitore eingebaut [JN10, S. 3ff] .

Nachdem das Verfahren erfolgreich durchgeführt wurde, kann zum Beispiel eine so genannte *Heatmap* (Abb.: 4.15) erstellt werden. Eine Heatmap zeigt, welche Bereiche eines Systems für längere Zeit oder häufiger betrachtet wurden. Dafür sind auf einer Heatmap verschiedene Farbbereiche zu erkennen. Je dunkler das Rot an einem Bereich auf der Heatmap dargestellt ist, desto häufiger wurde der bestimmte Bereich betrachtet. Daraus ergibt sich, dass ein blauer Bereich die Stellen markiert, die nicht so häufig beziehungsweise nur kurz betrachtet wurden [Zic09, S. 55f]. Somit ergeben sich für die Bewertung der Usability gute Möglichkeiten. Denn es kann unter anderem festgestellt werden, ob der Bildschirm überhaupt betrachtet wurde und welche Bereiche besondere Aufmerksamkeit der Probanden erhielten.

Abbildung 4.15: Abbildung einer Heatmap [Zic09, S. 57]

Weiterhin kann die Methode des Eye-Tracking gut mit weiteren Evaluationsmethoden kombiniert werden. So kann die Methode des Thinking-Aloud dazu genutzt werden, die Denkvorgänge des Probanden zu verbalisieren [Zic09, S. 56]. Oder durch einen abschließenden SUS-Fragebogen die Usability des Systems bewertet werden.

Jedoch birgt das Eye-Tracking Verfahren auch Nachteile. Zum Beispiel ist nicht immer eindeutig, ob der Proband das System beziehungsweise den betrachtenden Bereich auch tatsächlich wahrnimmt. Denn es kann durchaus sein, dass zwar ein Bereich betrachtet wird, allerdings über etwas ganz anderes nachgedacht wird. Weiterhin können unter Umständen Brillen und Kontaktlinsen die Erfassung von Daten verfälschen oder unmöglich machen [tea10].

4.4 Zusammenfassung

Diesem Kapitel ist zu entnehmen, dass das Usability Engineering ein bedeutender Faktor bei der Gestaltung einer gebrauchstauglichen Anwendung ist. Das Vorgehen nach dem User Centered Design bildet dabei einen Aspekt, der einen wesentlichen Teil innerhalb des Usability Engineering darstellt. Das nach diesem Modell beschriebene Vorgehen bezieht die Vorstellungen und Ideen der Endanwender stark in den Entwicklungsprozess mit ein, so dass sich eine einwandfreie Gebrauchstauglichkeit der Anwendung ergibt.

Zunächst wurden *Human Factors* im Kontext der visuellen Analyse näher betrachtet. Die Ergebnisse helfen dabei Visualisierungen so zu gestalten, dass Zusammenhänge von Daten möglichst einfach und natürlich von einem Menschen als Muster erkannt werden können und sollten beim Entwurf der Visualisierungen berücksichtigt werden. Die Gestaltgesetze sollten berücksichtigt werden, damit es nicht zu Fehlinterpretationen

kommt. Sie können aber auch ausgenutzt werden, indem zum Beispiel gleiche Werte gleich eingefärbt werden, damit sie nach dem Gesetz der Ähnlichkeit gruppiert empfunden werden. Es können auch verschiedene Beziehungen dargestellt werden. In Abbildung 4.16 sind zwei Gruppen dadurch gebildet, dass sie umschlossen sind (Gesetz der Region) und zwei Gruppen dadurch, dass sie sich ähnlich sind (Gesetz der Ähnlichkeit).

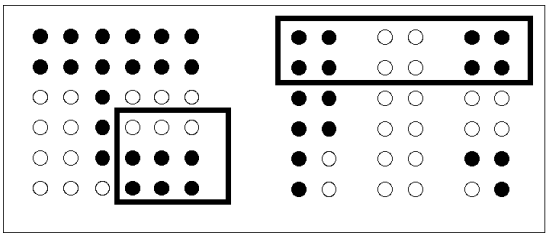


Abbildung 4.16: Beispiel für verschiedene Gestaltgesetze, die gleichzeitig wirken

Im darauffolgenden Abschnitt wurde beschrieben, wie Anforderungen nutzerorientiert ermittelt und geprüft werden können. Des Weiteren wurden einige Besonderheiten beschrieben, die zu beachten sind, wenn diese Studien in einem großen Unternehmen wie BMW stattfinden. Die Beschreibungen sind dabei nicht nur für die visuelle Analyse gültig.

Ein Vorgehen nach dem User Centered Design soll dabei helfen das System so zu entwerfen, dass die Anwender, die es benutzen, schnell lernen können, damit umzugehen und den bisherigen Analyseprozess zu verbessern. Die vorgestellten Evaluationsmethoden können dabei benutzt werden, um zu überprüfen, ob das System alle Anforderungen erfüllt.

Eine weitere Phase des User Centered Design ist die Evaluation. Unter Zuhilfenahme der in den Kapiteln beschrieben Evaluationsmethoden sind die gestalteten Visualisierungen unter anderem auf ihre Gebrauchstauglichkeit zu untersuchen. Sofern Verbesserungsmöglichkeiten gefunden wurden, können diese gegebenenfalls in der Anwendung angepasst werden.

Es zeigt sich also, dass es wichtig ist, die Wünsche und Bedürfnisse des Benutzers während der Entwicklung zu beachten. Des weiteren stellt sich heraus, dass man auf die Fähigkeiten und Beschränkungen der menschlichen Wahrnehmung berücksichtigen muss und ein gewisses Hintergrundwissen über Gestaltpsychologie äußerst hilfreich ist.

Kapitel 5

Designprozess - SCiVA

SCiVA (Surface Computing for Interactive Visual Applications) ist ein iterativer Designprozess zur Entwicklung von gestengesteuerten Anwendungen mit grafischer Oberfläche. Der Prozess ist in fünf Schritte unterteilt, die zyklisch durchlaufen werden. Diese Schritte sind im Einzelnen in der Reihenfolge der Durchführung:

1. Anforderungsanalyse
2. Visualisierungsfindung
3. Interaktionsdesign
4. Gesten
5. Evaluation

Nach dem Abschluss des letzten Schrittes (Evaluation), setzt der Prozess an dem Schritt wieder ein, an dem in der Evaluation Nachholbedarf festgestellt wurde (s. Abb. 5.1). Dieses Vorgehensmodell wird von Hesselmann, Boll und Heuten auf der EICS 2011 vorgestellt [HBH10]. Bei dem Prozess steht der Anwender und dessen Bedürfnisse im Vordergrund, es handelt sich um einen *User Centered Design* (UCD) Prozess.
Im Folgenden werden die einzelnen Schritte genauer erläutert.

5.1 Anforderungsanalyse

In der Anforderungsanalyse wird besonderes Augenmerk auf den Endbenutzer der zu entwickelnden Anwendung gelegt. Dabei müssen sich die Entwickler klar werden, welche Aufgaben und Ziele der Benutzer mit der Anwendungen erfüllen können soll. Hierfür ist zunächst ein tieferes Verständnis dafür notwendig, in welchem Zusammenhang der Benutzer die Anwendungen verwenden möchte (*Context of Use*). Nachdem der Anwendungskontext festgestellt wurde, muss darüber nachgedacht werden, welche Anforderungen die Anwendung erfüllen muss, um diese Aufgaben zu erfüllen. Zudem müssen bestimmte Einschränkungen in Betracht gezogen werden, die sich

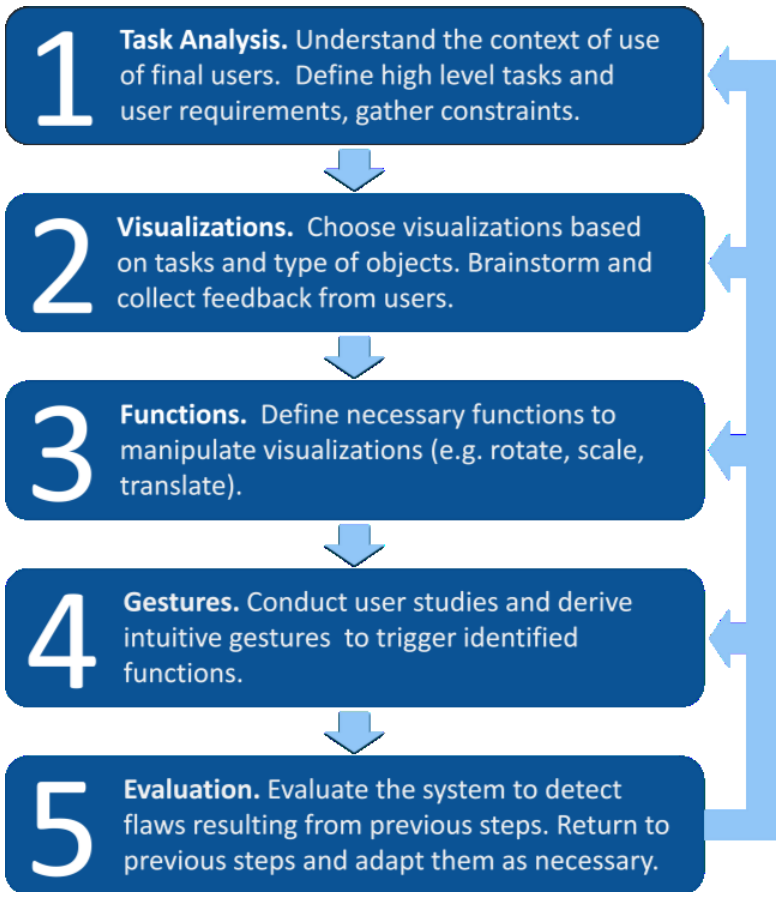


Abbildung 5.1: SCiVA-Prozess nach [HBH10]

beispielsweise aus der zur Verfügung stehenden Hardware ableiten lassen.
Erst wenn der Entwickler sich absolut sicher ist, dass er die Anforderungen des Benutzers korrekt erkannt und erfasst hat, sollte er zum nächsten Schritt gehen.

5.2 Visualisierungsfindung

Die Wahl der richtigen Visualisierung für ein gegebenes Problem ist ein wichtiger Aspekt der Entwicklung für gestenbasierte Anwendungen. Der Entwickler muss sich dazu ausführliche Gedanken machen, welche Visualisierungen in Frage kommen, um die gegebenen Daten zu demonstrieren und welche Visualisierungen technisch machbar sind. Dafür bietet es sich an, die Visualisierungsfindung in zwei Schritte zu unterteilen: Zunächst muss in Betracht gezogen werden, wie die Daten aufgebaut sind und was sie repräsentieren. Außerdem muss berücksichtigt werden, welche Ansprüche die Anwendung zu erfüllen hat. Handelt es sich beispielsweise um ein wissenschaftliches

Analysewerkzeug, werden vermutlich andere Visualisierungen benötigt, als für ein Spiel, obwohl eventuell sogar die selbe Datenbasis gegeben ist.
Im Anschluss sollte eine Gruppe von Leuten gefunden werden, die den Endbenutzer repräsentieren kann. Diese Gruppe hat dann die Aufgabe sich ihrerseits Gedanken zu der Problematik zu machen und sich Visualisierungen aus zu denken. Danach sollen sie ihre eigenen Ideen mit denen des Entwicklers vergleichen und mit ihm diskutieren, ob die erarbeiteten Visualisierungen zur Erfüllung der Aufgabe dienlich sein können und ob eventuell weitere Ideen gefunden werden können.
Es besteht auch die Möglichkeit der Repräsentantengruppe zuerst die vom Entwickler erzeugten Visualisierungen vor zu führen und danach der Gruppe die Aufgabe zu geben, eigene Ideen zu entwickeln. dabei besteht allerdings die Gefahr, dass die Gruppe von den Vorschlägen des Entwicklers beeinflusst ist und somit die Ergebnisse nicht das kreative Potenzial der Gruppe ausfüllen.
Während des gesamten Ablaufs sollte stets bedacht werden, dass die Anwendung mit Gesten und Fingern gesteuert werden soll, so dass keine Visualisierungen gewählt werden, die im angestrebten Szenario eher unpraktikabel sind. Viele der gebräuchlichen Visualisierungen sind darauf ausgelegt, durch Maus und Tastatur bedient zu werden. Bei gestengesteuerten Anwendungen muss jedoch berücksichtigt werden, dass durch die direkte Interaktion des Anwenders mit der Oberfläche des Bildschirmes eventuell bestimmte Teile des Sichtbereiches durch Finger, Hände oder Arme verdeckt werden könnten. Außerdem können Interaktionselemente durch Finger nicht so präzise verwendet werden, wie dies mit einer Maus möglich wäre.

5.3 Funktionen

Beim Interaktionsdesign muss sich der Entwickler in erster Linie Gedanken darüber machen, welche Funktionen ein Benutzer an der Anwendung ausführen können sollte. Dabei soll zunächst davon abstrahiert werden, dass die Bedienung über Berührung und Gesten vollzogen werden wird. Im Fokus der Betrachtung liegt allein, *was* der Anwender machen soll und nicht *wie* die Ausführung tatsächlich aussieht. Diese Trennung ist deswegen wichtig, damit die Entwickler nicht zu viele Funktionalitäten außer Acht lassen, nur weil diese gegebenenfalls auf den ersten Blick nicht oder nur schwer gestisch umsetzbar sind.
Auch in diesem Schritt soll der Benutzer im Mittelpunkt stehen und die entwickelten Funktionen sollten mit ihm abgestimmt werden, wobei hier die größte Wichtigkeit darin liegt, herauszufinden, ob alle vom Nutzer gewünschten Funktionen erkannt und erarbeitet worden sind, oder ob noch Nachholbedarf besteht.

5.4 Gesten/Interaktionsdesign

Nachdem im Interaktionsdesign die Funktionen der Anwendung herausgearbeitet worden sind, gilt es nun, passende Gesten dazu zu finden. Auch hier sollte enge Zusammenarbeit mit dem Benutzer erreicht werden. Dies kann zum Beispiel durch das von Wobbrock, Ringel Morris und Wilson auf der CHI 2009 vorgestellte Verfahren geschehen [WMW09a]. Hierbei werden dem Benutzer Bilder vom System gezeigt, wie es vor der Ausführung einer bestimmten Funktion aussieht, und wie es danach aussieht. Der Benutzer soll dann entscheiden, welche Geste sich wohl am besten für die Ausführung der Funktion eignet.
Hierbei wird es gegebenenfalls dazu kommen, dass eine bestimmte Geste zur Abdeckung verschiedener Funktionen bevorzugt wird. Hier gilt es eventuelle Uneindeutigkeiten zu beseitigen.

5.5 Evaluation

Der letzte Schritt des SCiVA-Prozesses ist die Evaluation. Hier steht das gesamte System auf dem Prüfstand. Die Evaluationsteilnehmer sollten daher einer für den Endbenutzer repräsentativen Personengruppe angehören. Die Evaluation hat die Aufgabe, eventuelle Fehler im Design und der Umsetzung aufzudecken, so dass diese korrigiert werden können, bevor die Anwendung praktisch eingesetzt wird. Bei der Evaluation sollte der Fokus auf qualitativer Ebene liegen, so dass die Teilnehmer beispielsweise dazu aufgefordert werden, während der Erprobung laut auszusprechen, was sie aktuell denken (sog. *Thinking aloud*-Verfahren). So kann der Entwickler die Gedankengänge der Benutzer besser nachvollziehen und auf dieser Basis Verbesserungen entwickeln.
Ein weiteres Ziel der Evaluation ist es, festzustellen, an welchen Stellen das System Schwachpunkte aufweist. Es könnte so beispielsweise herausgestellt werden, dass sich bestimmte Visualisierungen oder Gesten nicht oder nur schlecht dazu eignen, dass gegebene Problem abzubilden, beziehungsweise die gewünschte Funktion auszuführen. Je nachdem, auf welcher Ebene des SCiVA-Prozesses Fehler oder Schwachpunkte festgestellt wurden, setzt der Prozess an diesem Punkt wieder ein und alle nachfolgenden Schritte werden entsprechend angepasst.

5.6 Zusammenfassung

Durch die ständige Wiederholung der einzelnen Schritte anhand der Ergebnisse der Evaluation sorgt der SCiVA-Prozess dafür, dass sich die Anwendung während der Entwicklung ständig verbessert und immer besser an die Wünsche und Bedürfnisse

des Endbenutzers angepasst wird. Daraus resultiert eine Anwendung, die sowohl in funktionaler Hinsicht, als auch in der Gebrauchstauglichkeit eine hohe Qualität aufweisen kann. Durch die enge Zusammenarbeit mit dem Endbenutzer wird zudem verhindert, dass auf Grundlage einer eventuell unzureichenden Anforderungsdefinition eine Anwendung entwickelt wird, die nur sehr bedingt den Ansprüchen des Benutzer genügt. Zudem hat der Benutzer während der gesamten Entwicklung stets einen Einblick in die Fortschritte der Anwendung und kann aktiv am Entwicklungsprozess teilnehmen, so dass er auch sicher sein kann, dass das entwickelt wird, was er sich vorgestellt hat.
SCiVA ist ein iterativer Prozess zur Entwicklung von gestengesteuerten Anwendungen. Der iterative Charakter des Prozesses und die Fokussierung auf den Benutzer stellt ein gutes Vorgehensmodell für die Projektgruppe dar. Das liegt darin begründet, dass die von der Projektgruppe entstandene Anwendung ebenfalls nach den Grundsätzen des UCD entwickelt wurde und unter dem Banner des, ebenfalls auf Iterationen basierten, agilen Projektmanagements organisiert wurde.

Kapitel 6

KnoVA

Bei der Erstellung von interaktiven Anwendungen für die Datenexploration können drei Herausforderungen identifiziert werden: Die Integration heterogener Datenquellen zur Laufzeit, die Integration geeigneter Visualisierungen und die Anwendung von Wissen durch Domänen-Experten. In diesem Abschnitt wird das KnoVA (Knowledge-Based Visual Analytics) Referenzmodell vorgestellt, dass mit diesen Herausforderungen umgehen kann und zugleich die Erstellung von wissensgenerierenden Visual Analytics Anwendungen unterstützt [FHJA11].

6.1 Grundlagen

Bisher ist es bei der Analyse von Daten in der Praxis häufig so, dass diese in Data-Warehouse-Systemen gesammelt werden um anschließend Berichte nach vordefinierten Merkmalen generieren zu können [MRAK03]. Bei Domänen-Experten steigt jedoch das Verlangen nach dynamischen und interaktiven Lösungen, die auch Analysen bei Einzelfällen erlauben [FH10]. Gemeinsam mit diesen Domänen-Experten konnten drei Kernfaktoren für die Effektivität von Visual Analytics Anwendungen identifiziert werden [FHJA11]:

- **Datenintegration:** Zunächst müssen geeignete Informationen zur Behandlung einer bestimmten Analysefrage bereitgestellt werden. Die Herausforderung hierbei besteht in der Kombination verschiedener Datenquellen zu einem Datenbestand durch sinnvolles Daten-Mapping. Geografische Krankheitsmuster könnten beispielsweise mit der geografischen Bevölkerungsdichte kombiniert werden damit Gebiete mit verhältnismäßig hohem Krankheitsvorkommen erkannt werden können.
- **Visualisierungsintegration:** Der Datenbestand muss von der Analysesoftware durch geeignete Visualisierungen dargestellt werden. Oft ist es dabei nicht ausreichend nur eine Darstellung für eine bestimmte Analyseaufgabe anzubieten. Unterschiedlichste Visualisierungsmethoden sollten integriert werden, damit

Daten von verschiedenen Perspektiven und Detailtiefen betrachtet werden können. Im Regelfall nutzt ein Analyst unterschiedliche Visualisierungen zur selben Zeit. Jede Darstellung ist dabei für einen bestimmten Datenbereich geeignet. Für geografische Daten sind Kartendarstellungen meist sinnvoll wobei zeitbezogene Datenbereiche oft durch Punktdiagramme (siehe Kapitel 2.4.2) sehr gut dargestellt werden können.

- **Anwendung von Expertenwissen:** der dritte Einflussfaktor ist das Expertenwissen. Die Wahl der richtigen Kombination von Datenbereichen und geeigneten Visualisierungen sowie die Manipulation von Daten während des Explorationsprozesses werden durch Domänen-Experten verrichtet. Zur Unterstützung der Experten ist es wichtig, dass angemessene Interaktionen zur Verfügung stehen, die den Explorationsprozesses erleichtern.

Um mit diesen drei Herausforderungen umgehen zu können, wurde das KnoVA Referenzmodell entworfen.

Das Ziel des KnoVA-Ansatzes ist es, die Gewinnung von Expertenwissen zu ermöglichen, welches zuvor durch den Analysten im Analyseprozess angewendet wurde. Außerdem soll dieses Wissen in anderen Analysefragestellungen angewendet werden können. Um dies zu erreichen wurde eine Beschreibung auf Grundlage von bereits existierenden Klassifizierungsansätzen entwickelt. Das KnoVA-Vorgehen besteht aus vier unterschiedlichen Teilen [FHJA11]:

1. Ein Modell zur Beschreibung von Analyseanwendungen, das **KnoVA Referenzmodell**.
2. Ein Visualisierungs-Zustandsprozessmodell, welches auf einem angepassten „data-flow model“ [TIC09] basiert, wobei das KnoVA Referenzmodell zur Beschreibung der dort beschriebenen Zustände benutzt wird.
3. Eine Regel-Sprache, die auf dem KnoVA Referenzmodell basiert. Sie soll Regeln ausdrücken, mit denen Wissen abgeleitet werden kann.
4. Ein passender Algorithmus um anwendbares Wissen innerhalb bestimmter Analysesituationen zu identifizieren.

6.2 KnoVA Referenzmodell

Nach Keim [KFZ09] ist Visual Analytics ein iterativer Prozess mit drei unterschiedlichen Stufen: Datenauswahl und -vorbereitung, Visualisierung, Modellbildung. Eine Iteration führt zu Einsicht eines Analysten und dadurch zur Generierung von Wissen, welches im weiteren Analyseablauf genutzt werden kann. Dadurch kann Wissen, dass durch die

Erkenntnisse eines Analysten entsteht, in den Prozess zurückfließen. Dementsprechend ist Visual Analytics ein wissensgetriebener Prozess, in dem Expertenwissen implizit durch die Interaktionen des Nutzers angewandt wird. Daher kann eine Beschreibung der Systemzustände zusammen mit der Interaktion, die diesen Zustandswechsel ausgelöst hat, dazu verwendet werden, um das angewandte Wissen implizit zu beschreiben. Dementsprechend ist das Ziel des KnoVA Referenzmodells, eine Beschreibung der Systemzustände und Interaktionen zu ermöglichen [FHJA11]. Das KnoVA Referenzmodell basiert auf der von Keim [Kei01] vorgeschlagenen Klassifizierung, die bereits in Kapitel 2.3 vorgestellt wurde. Diese Klassifizierung wurde jedoch bedeutend erweitert und umfasst 32 klassifizierende Eigenschaften. Diese sind in sechs Klassen eingeteilt, die wiederum einige Unterklassen enthalten. In Abbildung 6.1 wird das Modell in Form des UML-Paket-Diagramms verdeutlicht. Die sechs unterschiedlichen Klassen werden im Folgenden beschrieben [FHJA11]:

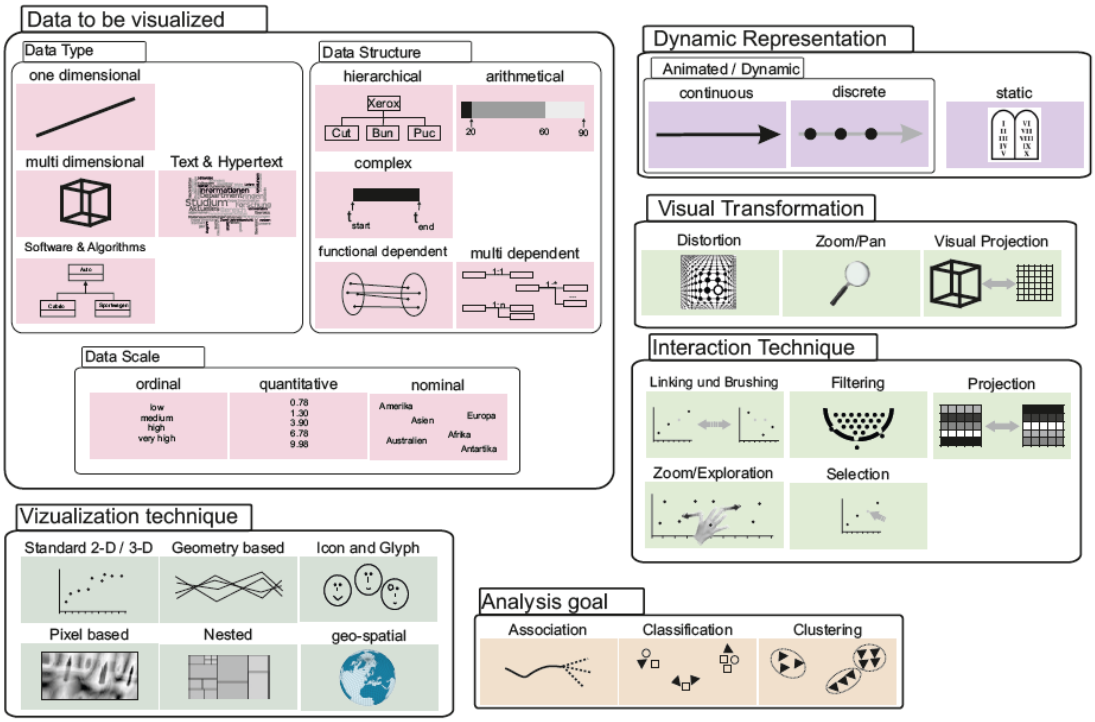


Abbildung 6.1: UML-Darstellung des KnoVA Referenzmodells [FHJA11]

Data to be visualized: Diese Klasse klassifiziert die Daten, die von einem Visual Analytics System genutzt werden. Innerhalb dieser Klasse gibt es drei Unterklassen: der Datentyp (z.B. ein- oder multidimensional), die Datenstruktur (z.B. hierarchisch) und die Werte, welche die Daten aufweisen (z.B. ordinale oder quantitative Daten)

Analysis Goal: Die meisten Analysewerkzeuge sind für ein spezielles Analyseziel optimiert, genauso wie die meisten Visualisierungen. Auch wenn viele Visualisierungen für unterschiedliche Ziele angewendet werden können ist es sinnvoll, alle Analyseziele die Anwendbar sind, zu identifizieren.

Visual Transformation: Diese Klasse fasst Veränderungen an den Visualisierungen zusammen, die nur die Darstellung, nicht aber den Zustand der betroffenen Daten verändern. Zu dieser Klasse gehört die „Distortion“, die bereits in Abbildung 2.9 dargestellt wurde. Typischerweise sind Interaktionen notwendig um diese Techniken zu nutzen.

Interaction Technique: In dieser Klasse sind Techniken der Interaktion gruppiert, die zu Veränderungen der Daten führen. Diese Techniken unterscheiden sich von den visuellen Transformationen insofern, dass sie nicht nur die Visualisierung, sondern auch den aktuellen Systemzustand verändern. Zum Beispiel können hierarchische Daten bei einem hineinzoomen detaillierter repräsentiert werden, als es vor dem Zoomen der Fall war. Wenn zuvor nur die Daten der ersten Hierarchieebenen berücksichtigt wurden, werden nach dem Zoomen somit die tiefer geschachtelten Daten verwendet.

Dynamic Representation: Visualisierungsmethoden können, solange keine Nutzereingaben geschehen, in statische oder animierte Darstellungen unterteilt werden. Animationen können dabei entweder kontinuierlich mit glatten Zustandsübergängen oder diskret (wie z.B. eine Diashow) verlaufen.

Visualization Technique: In dieser Klasse werden die verschiedenen Visualisierungstechniken zusammengefasst. Jede Visualisierung repräsentiert Daten auf eine andere Art. Einige hiervon wurden bereits in Kapitel 2.3.2 erörtert.

6.3 Zusammenfassung

Das KnoVA Referenzmodell basiert auf den Arbeiten von Keim [Kei01], in denen er eine Klassifizierung für visuelle Analyseanwendungen (Kapitel 2.3) vornimmt. Diese Klassifizierung wurde jedoch durch weitere Klassen, Unterklassen sowie klassifizierende Eigenschaften bedeutend erweitert. Im Gegensatz zu Keims Ansatz wird das KnoVA Referenzmodell vielmehr in einer deskriptiven Weise benutzt. Die modellgetriebene Herangehensweise von KnoVA unterstützt die Entwicklung von mächtigen visuellen Analyseanwendungen. Aus diesem Grunde geschah die Umsetzung des TOAD-Projektes unter Berücksichtigung dieses Aspektes. Die Realisierung auf Basis des KnoVA Referenzmodells sowie die wissensbasierten Funktionen des TOAD-Systems werden in Kapitel 13.4 ausführlich beschrieben.

Kapitel 7

BMW Datalogs

Dieses Kapitel stellt drei verschiedene Prototypen der BMW Group für Forschung und Technik zur Unterstützung der Ingenieure im Bereich der In-Car-Kommunikation vor. Für das grundlegende Verständnis werden hierfür im ersten Unterkapitel die Grundlagen der In-Car-Kommunikation vorgestellt. Der Schwerpunkt liegt hierbei auf dem physischen und funktionalen Netzwerks eines Automobils. Die einzelnen Prototypen werden im zweiten Unterkapitel vorgestellt. Abschließend werden im letzten Unterkapitel die Prototypen evaluiert.

7.1 Grundlagen der In-Car-Kommunikation

Der Funktionsumfang von modernen Automobilen wächst heutzutage stetig an. Das Ziel der neuen Funktionalitäten, wie beispielsweise einem integrierten Navigationssystem, ist ein komfortableres und sicheres Fahren für den Fahrer.[Sed09] Einhergehend mit dem steigenden Funktionsumfang ist eine steigende Komplexität der zugrundeliegenden Kommunikationsarchitektur. Für die Suche und Analyse von potentiellen Quellen und Auswirkungen von Fehlern innerhalb dieser Kommunikationsarchitektur benötigen die zuständigen Ingenieure einen leicht verständlichen und gleichzeitig detaillierten Einblick in die Kommunikationsprozesse und ihren Kontext. Die aktuellen Tools zur Unterstützung der Ingenieure sind größtenteils textbasiert und können keine Korrelationen zwischen den Daten innerhalb der Kommunikationsarchitektur aufzeigen. [SKHB09]

Die Kommunikationsarchitektur eines Automobils basiert auf einem physischen und einem funktionalen Netzwerk, welches vor allem in modernen Automobilen sehr komplex ist (vgl. Abb. 7.1).

Das funktionale Netzwerk besteht aus einer Vielzahl an Functional Blocks (FB). Ein FB ist eine logische Softwareeinheit, welche für eine spezielle Aufgabe zuständig ist. Durch das Zusammenspiel mehrerer funktionaler Blöcke können auch komplexe Funktionen, wie beispielsweise die Steuerung einer Klimaanlage, realisiert werden. Die funktionalen

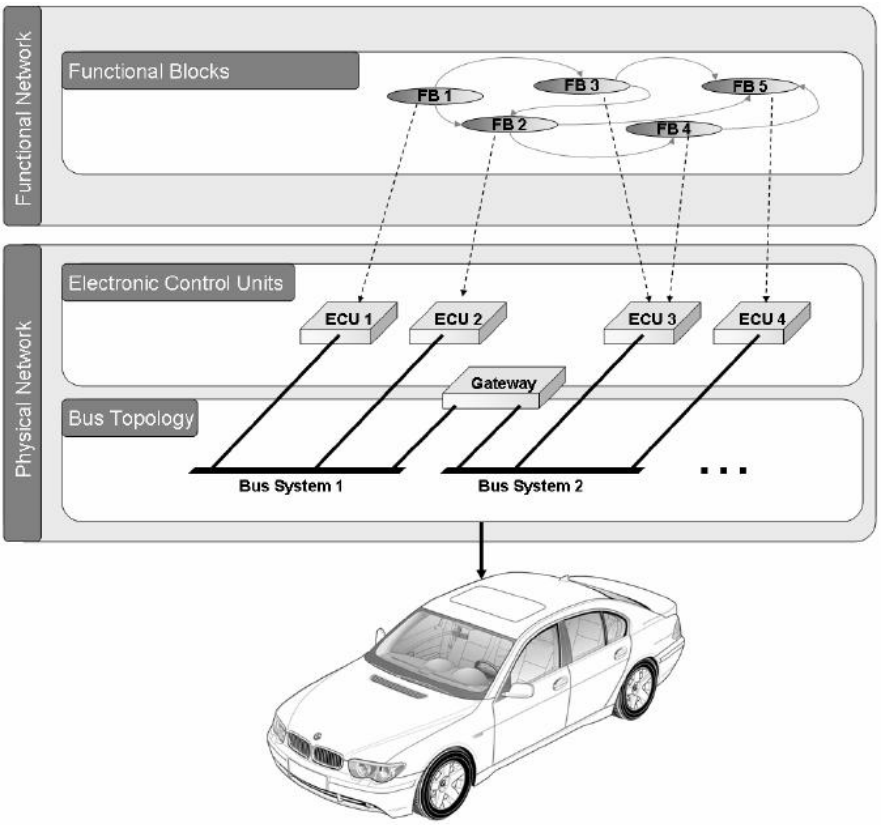


Abbildung 7.1: Physisches und funktionales Netzwerk eines Automobils [SHS+08]

Blöcke werden mit Hilfe von Electronic Control Units (ECU) implementiert. Moderne Automobile besitzen bis zu 70 ECU, welche verteilt auf das komplette Automobil verbaut sind. Die einzelnen ECU sind über Bus-Systeme miteinander verbunden. Ein Bus-System dient als Träger für den Informationsaustausch. Bekannteste Beispiele für Bus-Systeme sind das Controller Area Network (CAN), das Media Oriented Systems Transport (MOST) und FlexRay. Die einzelnen Bus-Systeme sind über einen Gateway miteinander verbunden und können somit miteinander kommunizieren. Das Zusammenspiel aus ECU, Bus-System und Gateway wird als physisches Netzwerk bezeichnet. [SHS+08]

In modernen Automobilen werden über das physische Netzwerk bis zu 1.000.000 Nachrichten pro Minute verschickt. Eine Nachricht enthält hierbei den sendenden funktionalen Block mit dem zugehörigen ECU, verschiedene gekapselte Signale und weitere Detailinformationen, wie beispielsweise einen Zeitstempel. Die Beschreibung der Kommunikationsprozesse innerhalb des Kommunikationsnetzwerkes basiert auf textuellen Informationen. Allein auf Grundlage dieser textuellen Informationen benötigen die Ingenieure viel Zeit, Erfahrung und Fachwissen, um Korrelationen und Probleme bezüglich der In-Car-Kommunikation zu identifizieren und daraus Aktivitäten

abzuleiten. [Sed08a]

Die aktuellen Tools im Bereich der In-Car-Kommunikation sind größtenteils textbasiert. Der De-Facto-Standard für die Fehler- und Netzwerkanalyse innerhalb der Kommunikationsarchitektur ist CANalyzer. CANalyzer ist Bestandteil einer Toolkette, welche sich mit Bus-Systemen und Kommunikationsnetzwerken innerhalb von Automobilen beschäftigt. Das Tool bietet neben einer hohen Funktionalität nur rudimentäre Visualisierungen über Liniendiagramme und Fortschrittsbalken und könnte somit über interaktive Informationsvisualisierungen stark verbessert werden. [SKHB09]

Die folgenden Prototypen der BMW Group für Forschung und Technik wurden auf der Forschungsgrundlage der Informationsvisualisierung (IV) entwickelt und versuchen genau diese Lücke zu schließen.

7.2 Prototypen

Dual-View Visualization mit ECU-Map und MSCar

Der erste Prototyp bietet die Möglichkeit fehlerhafte Datenaufzeichnungen (sogenannte Data Traces) in die Anwendung zu importieren und diese interaktiv zu untersuchen. Die Analyse dieser fehlerhaften Data Traces bedingt die Analyse aller möglichen funktionalen Abhängigkeitsketten. Dies bedeutet, dass neben dem auslösenden FB auch alle Vorgänger und deren Vorgänger sowie alle Nachfolger und deren Nachfolger untersucht werden müssen. Beispielsweise kann ein gemeldeter Fehler bereits durch einen vorangegangenen FB verursacht worden sein und weitere Auswirkungen für die jeweiligen Nachfolger haben (vgl. Abb. 7.2).

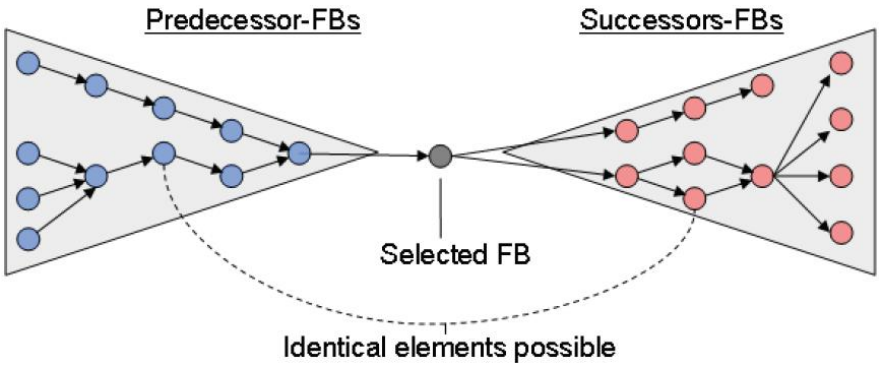


Abbildung 7.2: Vorgänger/Nachfolger-Beziehungen von Funktionsblöcken

Der Prototyp ist ein vertikal getrennter Dual View Ansatz mit einem kompletten Abbild des physischen Netzwerks (ECU-Map) in der linken Sicht und einem dynamisch

adaptierten und situationsspezifischen Auszug des funktionalen Netzwerks (MSCar) in der rechten Sicht (vgl. Abb. 7.3). Dieser Ansatz entspricht dem Prinzip der Multiple Coordinated Views (MCV).[Rob07]

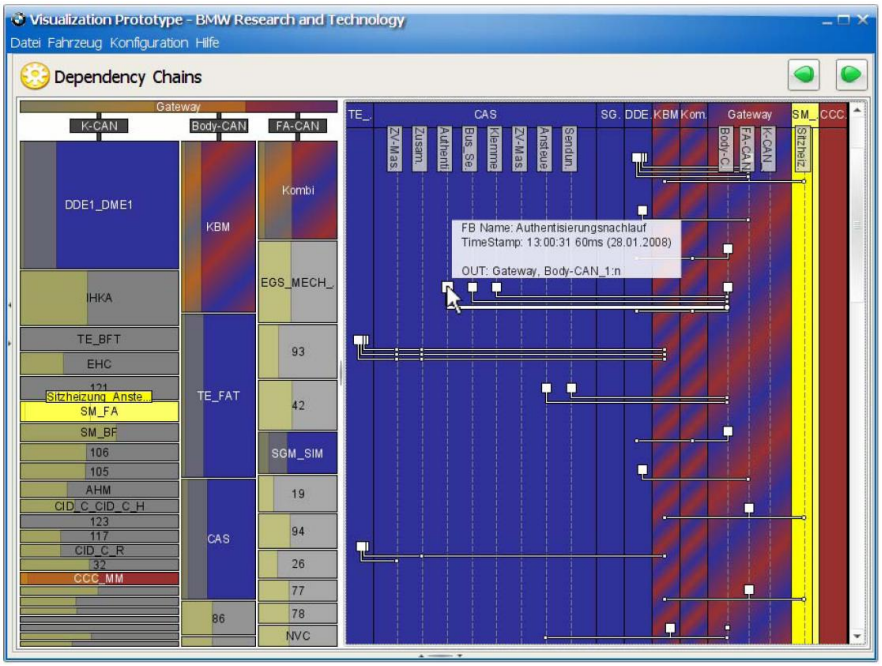


Abbildung 7.3: Dual-View Visualization mt ECU-Map und MSCar

Die ECU-Map bildet das physische Netzwerk durch eine Mischung aus Strukturdiagramm und Tree Map [JS91] ab. Die einzelnen Bus-Systeme werden als Spalten dargestellt und sind über den Gateway miteinander verbunden. Die Breite der Spalte richtet sich nach den zugehörigen FB im Verhältnis zur Gesamtheit aller FB. Unter jeder Spalte sind die zum Bus-System zugehörigen ECU visualisiert. Die Höhe von jeder ECU richtet sich nach den zugehörigen FB im Verhältnis zu den FB des Bus-Systems. Weiterhin besitzt jede ECU ein gelbes Overlay, welches den Anteil der derzeitig aktiven und inaktiven FB darstellt.

Durch die Auswahl eines ECU in der ECU-Map werden alle aktiven und inaktiven FB separat in alphabetischer Reihenfolge angezeigt. Zusätzlich werden zu allen aktiven FB die Anzahl der direkten Vorgänger und Nachfolger angezeigt. Ein aktiver FB kann nun ausgewählt werden und beide Sichten des Prototyps werden simultan aktualisiert. In der ECU-Map ändern sich die Farbkodierungen wie folgt [SHS+08]:

- Die ausgewählte ECU wird hellgelb
- Alle ECU mit einem oder mehreren FB, die in der Abhängigkeitskette als Vorgänger auftreten, werden dunkelblau

- Alle ECU mit einem oder mehreren FB, die in der Abhängigkeitskette als Nachfolger auftreten, werden rot
- Alle ECU mit einem oder mehreren FB, die in Abhängigkeitskette sowohl als Vorgänger als auch als Nachfolger auftreten, werden rot-blau-gestreift

Die rechte Sicht des Prototyps (MSCar) wird erstmalig durch Auswahl eines FB in der ECU-Map angezeigt. MSCar stellt eine Erweiterung von großskalierten Message Sequence Charts (MSC) dar und repräsentiert die funktionale Abhängigkeitskette des ausgewählten FB. Die beteiligten ECU werden als horizontale Rechtecke am oberen Rand des Bildschirms angezeigt und sind farblich analog zur ECU-Map kodiert. Die Breite eines jeden ECU richtet sich nach der Anzahl der beteiligten FB. Alle Kommunikationsverbindungen in denen der ausgewählte FB als Sender und/oder Empfänger einer Nachricht auftritt, werden als horizontale Linien in chronologischer Reihenfolge visualisiert. Auf der Kommunikationslinie wird der Empfänger einer Nachricht mit einem kleinen Punkt und den Sender mit einem kleinen Quadrat markiert.

Folgende Funktionen werden derzeit von MSCar unterstützt:

- Dynamic Path Highlighting: Der Benutzer kann einzelne Nachrichten innerhalb von MSCar markieren und die Kommunikationslinie wird daraufhin breiter dargestellt. Dieses Feature ist vor allem dann nützlich, wenn durch parallele Nachrichten an verschiedene Empfänger die Komplexität des MSC stark ansteigt.
- Semantic Zoom: MSCar unterscheidet zwischen dem Fokus und dem Kontext Modus (vgl. Abb. 7.4). Im Fokus Modus werden zusätzlich zum ECU auch die zugehörigen FB als gestrichelte Linien angezeigt. Der Fokus Modus basiert also auf funktionalen Abhängigkeiten. Der Kontext Modus basiert auf physischen Abhängigkeiten und es werden somit keine FB angezeigt. Folglich kann der Benutzer beim Fokus Modus im Gegensatz zum Kontext Modus neben dem ECU auch den sendenden oder empfangenden FB erkennen.

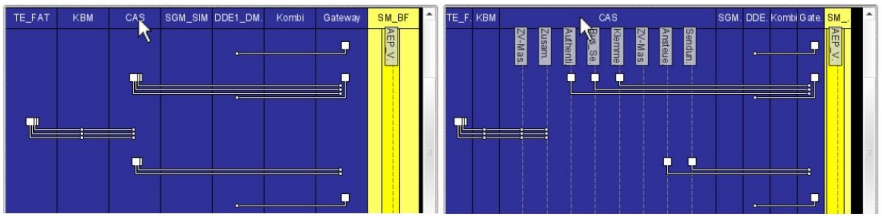


Abbildung 7.4: Kontext und Fokus Modus in MSCar [SHS+08]

- Details On Demand: Wenn der Benutzer den Mauszeiger auf einen Empfänger oder einem Sender einer Nachricht platziert, erscheint eine semitransparente Box (Hover) mit weiteren Detailinformationen. Neben dem Namen des ECU und

des FB, enthält der Hover weitere Detailinformationen, wie beispielsweise den exakten Zeitpunkt der Aktivität.

- History: MSCar speichert jede angezeigte Abhängigkeitskette in einer History. Der Benutzer kann somit einzelne Aktionen rückgängig machen oder schnell auf frühere Abhängigkeitsketten zugreifen.

Der vorgestellte Prototyp wurde in Java mit dem Piccolo Framework implementiert.[Sed08b]

CarComViz

Die Daten im Bereich der Informationsvisualisierung sind häufig nicht rein abstrakt, sondern weisen zusätzlich einen Bezug zur Realwelt auf. Traditionelle Ansätze sind jedoch meist ausschließlich für die Visualisierung von abstrakten Daten (InfoVis-Ansätze) oder für die Visualisierung von Daten mit Bezug zur Realwelt (SciVis-Ansätze) konzipiert.

Die Daten im Bereich der In-Car-Kommunikation stellen ein gutes Beispiel für hybride Daten dar. Sie repräsentieren zum einen abstrakte Signale und Nachrichten, welche zwischen Funktionsblöcken gesendet werden und zum Anderen sind sie über den zugehörigen ECU mit einer Position in der Realwelt verbunden. Weiterhin können die Signale und Nachrichten zu physischen Effekten, wie beispielsweise dem Öffnen eines Fensters führen.

CarComViz ist ein weiterer Prototyp zur Visualisierung von Daten im Bereich der In-Car-Kommunikation. Der Prototyp versucht das Verständnis für die semantische Verbindung zwischen abstrakten Daten und Objekten der Realwelt zu verbessern. Bei CarComViz handelt es sich um einen InfoVis-Ansatz, welcher zusätzlich um ein 3D-Modell innerhalb eines MCV-Systems angereichert wird. Durch die Verwendung eines MCV-Systems kann der Prototyp die Komplexität der Daten teilen. Der Benutzer kann auf visuelle Art die einzelnen Nachrichten und ihre Detailinformationen untersuchen und gleichzeitig werden dem Benutzer der physische Zustand des Automobils bzw. die Auswirkungen von einzelnen Funktionen in einem 3D-Modell angezeigt.

Das Design von CarComViz beruht auf den drei Sichten Autobahn View, List View und 3D Model View (vgl. Abb. 7.5). Die Autobahn View ist hierbei die zentrale Sicht zur Visualisierung der abstrakten Daten. Der Aufbau der Sicht entspricht der Metapher einer dicht befahrenen Autobahn. Die Bus-Systeme werden hierbei als separierter, übergeordneter Block visualisiert (der Autobahn). Jedes Bus-System transportiert die Nachrichten und Signale der zugehörigen ECU. Die einzelnen ECU werden als horizontale Balken innerhalb des Bus-Systems angezeigt (die Spuren der Autobahn). Jeder horizontale Balken enthält eine Vielzahl an schwarzen Rechtecken, welche die einzelnen Nachrichten von oder an einen ECU repräsentieren (die Autos). Zusätzlich

enthält die Sicht eine Zeitachse, um die genauen zeitlichen Abstände zwischen den Nachrichten darstellen zu können. Im Vergleich zum ersten Prototyp werden mit Hilfe der Autobahn View also nicht nur die kausalen Zusammenhänge, sondern auch die genauen zeitlichen Abhängigkeiten dargestellt. Um einzelne Nachrichten dauerhaft hervorzuheben, kann der Benutzer einen sogenannten Point of Interest (POI) setzen. Der POI wird dem Benutzer als kleiner roter Kreis oberhalb der Nachricht dargestellt. Die Navigation innerhalb der Autobahn View erfolgt über Zooming (zeitlich genauer oder zeitlich grober) und Panning (zeitlich vor und zeitlich zurück).

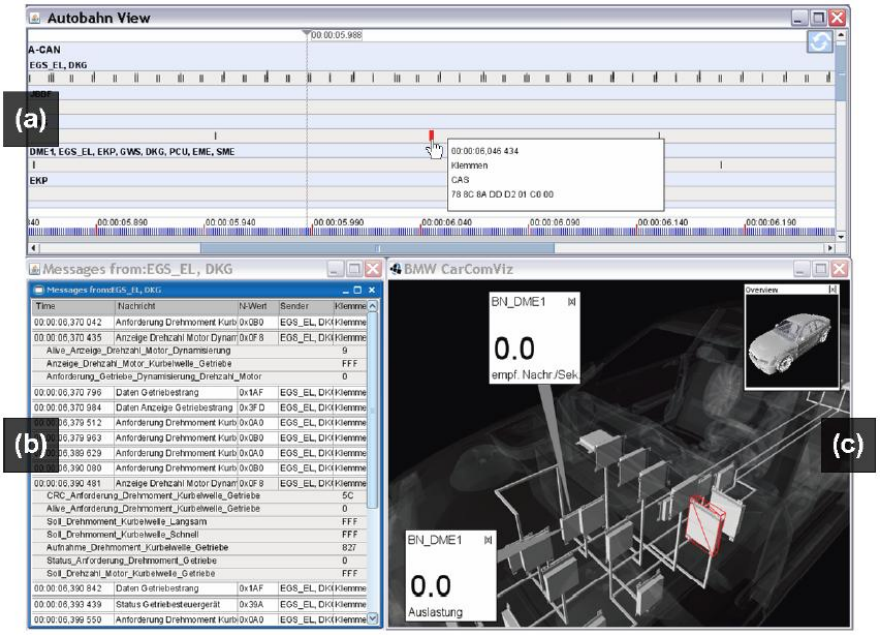


Abbildung 7.5: CarComViz [SRH+09]

Die zweite Sicht des Prototyps ist die List View. Der Benutzer kann mit der Maus eine oder mehrere Nachrichten in der Autobahn View auswählen und ihm werden Detailinformationen zu den ausgewählten Nachrichten in textueller Form innerhalb der List View angezeigt. Zu den angezeigten Informationen gehören die genauen Zeiten, gekapselte Signale oder hexadezimale Rohdaten der Nachrichten. Die List View wurde im Prototyp verwendet, da die textuelle Darstellung die beste Möglichkeit ist, um Detailinformationen darzustellen.

Die 3D Model View ist die dritte Sicht von CarComViz. Sie zeigt ein 3D-Modell des zu untersuchenden Automobils. Mit Hilfe des 3D-Modells kann reales Verhalten, wie beispielsweise das Öffnen eines Fensters, über eine Animation dargestellt werden. Die Animationen werden vom Benutzer durch Navigation innerhalb der Autobahn View ausgelöst. Die 3D Model View verwendet zwei Perspektiven, welche durch einen „picture in picture“-Ansatz realisiert sind. Die Außenperspektive des Automobils wird

für die Anzeige der bereits genannten Animationen verwendet. Die Innenperspektive zeigt die innere Architektur des Automobils an und beispielsweise wird in ihr bei der Auswahl einer Nachricht in der Autobahn View die Position des zugehörigen ECU hervorgehoben. Zusätzlich zur Hervorhebung kann sich der Benutzer semitransparente 2D-Boxen zu den einzelnen Komponenten anzeigen lassen. Es können sowohl mechanische Komponenten als auch funktionale Komponenten angereichert werden. Die einzelnen Boxen zeigen benutzerspezifische Informationen, wie beispielsweise die Geschwindigkeit des Automobils oder den Ladestatus eines Bus-Systems. Beim Zusammenspiel der einzelnen Sichten verwendet CarComViz den Equal View-Ansatz. Dies bedeutet, dass die einzelnen Sichten des Prototyps jeweils den gleichen Sachverhalt darstellen. Die Sichten agieren also als parallele Sichten innerhalb eines MCV-Systems. Der Vorteil des Equal View-Ansatzes ist, dass weder ein Fokus auf die abstrakten Daten noch ein Fokus auf das 3D-Modell gesetzt wird. Der Benutzer kann mit den abstrakten Informationen arbeiten und das 3D-Modell für das tiefere Verständnis benutzen und umgekehrt. Die Verbindung der Sichten erfolgt entweder über Item-Linking oder über Semantic-Linking. Unter Item-Linking versteht man die Hervorhebung von Daten über sogenannte Items. Ein Beispiel hierfür ist die Hervorhebung eines ECU in der Innenperspektive des Automobils bei Auswahl einer Nachricht in der Autobahn View. Beim Semantic-Linking führt die Auswahl oder Navigation innerhalb der Autobahn View zu tatsächlichem Verhalten in der Außenperspektive des 3D-Modells. Ein Beispiel hierfür ist das Anfahren des Automobils, wenn bestimmte Nachrichten in der Autobahn View markiert werden. Es können somit ganze Aktivitäten über Animationen und auch der Status eines Objektes dargestellt werden.

CarComViz wurde, wie auch der erste Prototyp, in Java mit dem Piccolo Framework programmiert. [SRH+09]

MostVis

Der MOST-Bus (Media Oriented Systems Transport) dient dem Transport von Audio-, Video-, Sprach- und Datensignalen und wird für Multimedia-Anwendungen in einem Automobil verwendet. Die physische und die höhere funktionale Schicht (Adressierung etc.) wurden durch die MOST Cooperation standardisiert. Die einzelnen Funktionen des Bus-Systems werden jedoch vom jeweiligen Hardwarehersteller selbst programmiert. Als Informationsquelle dient den Entwicklern hierbei der MOST-Funktionskatalog. Er enthält die textuellen Beschreibungen für derzeit mehr als 3500 Funktionen. Zugang zum Funktionskatalog erhalten die Entwickler über ein textuelles Datenbankinterface, ein PDF-Dokument oder eine XML-Version des Katalogs. Das textuelle Datenbankinterface wird hauptsächlich für die die Erstellung neuer Funktionen und die Bearbeitung aktueller Funktionen benutzt. Das PDF-Dokument hat gegenwärtig einen Umfang von ca. 4000

Seiten und dient sowohl als Anforderung für die Gerätehersteller als auch als schnelle Referenz für MOST-Nichtexperten.

Das Ziel von MostVis ist die bessere Unterstützung bei der Suche im Funktionskatalog für MOST-Nichtexperten und ein zusätzlicher Support bei der erweiterten Arbeit von MOST-Experten. Die große Herausforderung hierbei ist die Bewältigung von großen Datenmengen und das Design von adäquaten, interaktiven Untersuchungstechniken. Der Prototyp basiert, wie auch die ersten beiden Prototypen, auf interaktiven Multiple Coordinated Views und zeigt die zugrundeliegende Hierarchie des Funktionskatalogs als Nodelinks (Knoten, welche gleichzeitig auch Links sind). Der Funktionskatalog kann mit Hilfe eines XML-Exports der Datenbank in den Prototypen importiert werden.

Die Hierarchie wird direkt aus der XML-Datei übernommen und baut sich wie folgt auf:

1. Ebene: Automobilvariante
2. Ebene: Electronic Control Unit (ECU)
3. Ebene: Functional Block (FB)
4. Ebene: Funktionen des FB
5. Ebene: Weitere Parameter der Funktion

Das Design von MostVis basiert auf Shneidermans Mantra zur Informationsvisualisierung „Overview first, zoom and filter, details on demand“. Der Prototyp unterscheidet zwischen zwei verschiedenen Hauptsichten. Beide Sichten visualisieren den gleichen Sachverhalt in unterschiedlichem Kontext.

Die Local View (vgl. Abb. 7.6) ist ein leicht zu navigierender, horizontaler Baum. Beim Start des Prototyps zeigt der Baum mit der Automobilvariante und den zugehörigen ECU die ersten beiden Stufen der Hierarchie an. Durch Auswahl eines Knoten kann der Baum erweitert werden. Jeder Knoten enthält neben seiner Beschriftung ein zusätzliches Icon, welches seinen Elementtyp visualisiert.

Die Expanded View (vgl. Abb. 7.7) ermöglicht einen Überblick über den gesamten Funktionskatalog und dient zur Untersuchung von Elementen in ihrem globalen Kontext. Der Funktionskatalog wird als Hyperbolic 2D Tree[LR96] visualisiert. Da die Komplexität durch die große Datenmenge die menschliche Wahrnehmungsfähigkeit überschreitet, kann ein Filter die angezeigte Menge an Hierarchieebenen reduzieren. Zusätzlich kann der Anwender eine weitere Sicht (Zoom View) benutzen, da auch die Filterdarstellung noch sehr komplex sein kann.

Zur weiteren Arbeit mit dem Funktionskatalog wurden einige zusätzliche Funktionalitäten in den Prototypen implementiert. Die Suche erlaubt es dem Benutzer im kompletten Funktionskatalog, in einem Unterbaum oder nach speziellen Datentypen zu suchen. Das Ergebnis der Suche wird dem Benutzer als textuelle Darstellung über eine List View

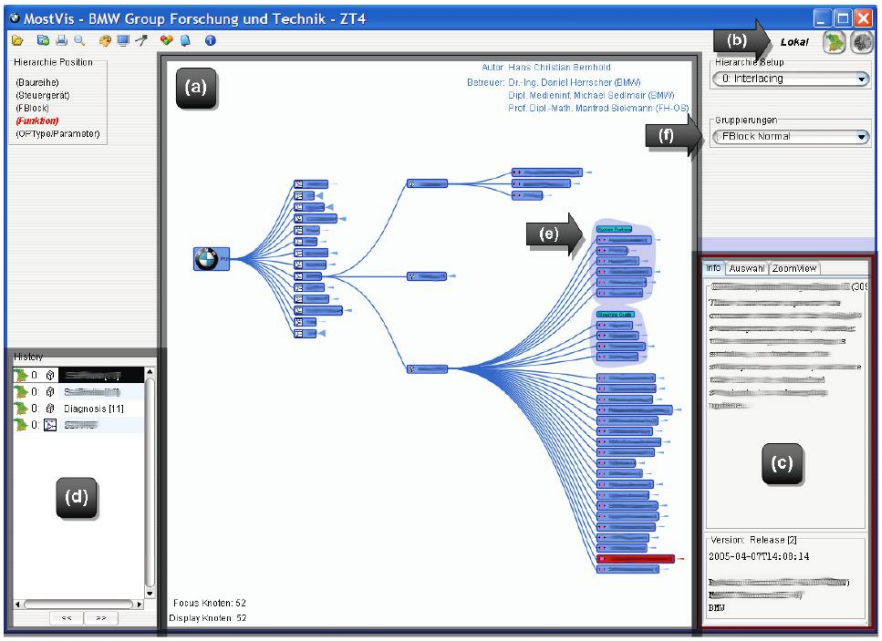


Abbildung 7.6: MostVis - Local View [SBH+09]

dargestellt. Zusätzlich zur List View werden die Suchergebnisse in der aktuellen Sicht des Funktionskatalogs farblich hervorgehoben. Die Gruppierung von Elementen stellt eine weitere Funktion von MostVis dar und ermöglicht es dem Benutzer bestimmte Elemente dynamisch zu clustern. Die Gruppierungsmuster können zum Einen bereits beim Import der Daten in der XMLDatei enthalten sein und können zum Anderen auch manuell nachgepflegt werden. Eine zusätzliche Funktionalität ist die Historisierung. Bei der Arbeit mit MostVis wird jeder Ausschnitt des Funktionskatalogs (z.B. das Ergebnis einer Suche) gespeichert und kann über Vor- und Zurück-Buttons sowie über eine direkte Auswahl angezeigt werden. Zuletzt sei noch die Möglichkeit der Datenausgabe erwähnt. MostVis ermöglicht dem Benutzer den Druck der aktuellen Sicht oder der Export in eine Excel- oder Bild-Datei.

MostVis wurde in Java mit dem Prefuse-Framework programmiert. [SBH+09]

7.3 Evaluation

Alle vorgestellten Prototypen wurden durch eine Benutzerstudie evaluiert. Die Teilnehmer der User Study waren jeweils entweder Experten und Nicht-Experten im Bereich der In- Car-Kommunikation. Die Durchführung der User Studies gliederte sich in drei Phasen. Zunächst haben die Teilnehmer eine Einführung bekommen. Die Einführung wurde teilweise videounterstützt durchgeführt und dauerte ca. 30 Minuten. Nach der

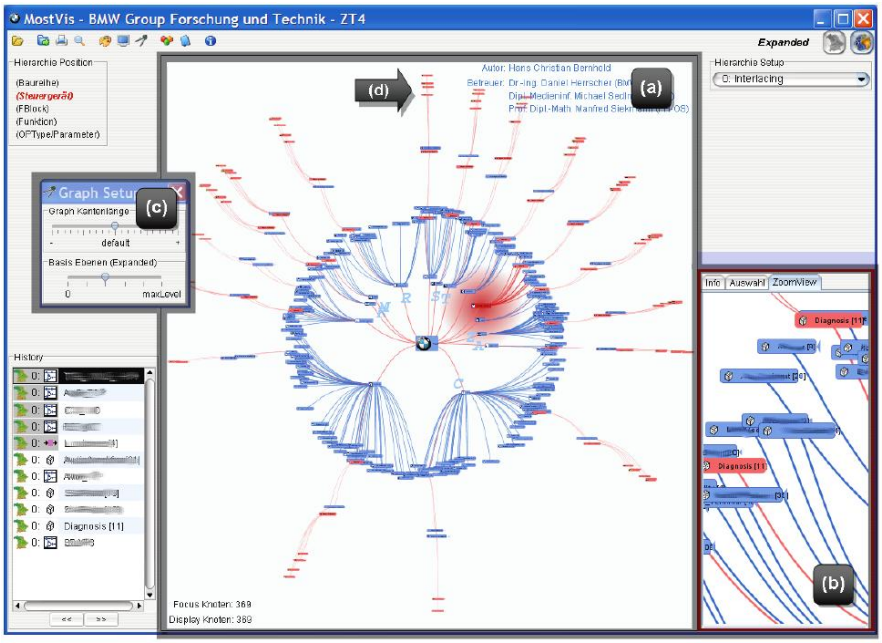


Abbildung 7.7: MostVis - Expanded View [SBH+09]

Einleitung mussten die Teilnehmer verschiedene Aufgaben mit den Prototypen lösen und es wurde die Zeit zur Durchführung der Aufgabe gemessen. Zum Abschluss der Evaluation musste jeder Teilnehmer einen Fragebogen ausfüllen und es wurde ein Interview für zusätzliches Feedback durchgeführt. [SKHB09]

Die Teilnehmer der User Study bewerteten die einzelnen Prototypen im Rahmen der Evaluation allesamt als sehr positiv. Bei allen drei Prototypen wurde die Zeitersparnis durch Benutzung der Prototypen im Vergleich zu den aktuell eingesetzten Tools hervorgehoben. Kritikpunkte an den einzelnen Prototypen wurden insgesamt nur sehr selten geäußert. Beispielsweise lassen sich in der Autobahn View mehrere Nachrichten durch Ziehen eines Rechtecks auswählen und die Benutzer erwarteten hier eine Zoom-Funktion. [SKHB09]

Teil II

Projektorganisation

Kapitel 8

Entwicklung Vorgehensmodell Projektmanagement

Klassische Projektmanagement Methoden, wie das Wasserfallmodell oder das V-Modell, gibt es schon seit vielen Jahren und haben sich schon in einigen Projekten bewährt. So muss man sich die Frage stellen, worin die Motivation liegt neue Projektmanagement Methoden zu nutzen und seine vorhandenen Strukturen und Gewohnheiten umzustellen. Der Wandel vom klassischen Projektmanagement zum agilen Projektmanagement im Bereich der Softwareentwicklung ist in den letzten Jahren häufiger zu beobachten. So wechseln nicht nur experimentell arbeitende Softwareteams zu den agilen Methoden. Dies begründet sich unter anderem in folgenden Thesen:

- Die hohe Innovationsgeschwindigkeit in der IT-Branche bringt einerseits kurze Produktlebenszyklen mit sich und somit auch möglichst kurze und flexible Entwicklungszeiten.
- Kunden sind nicht mehr in der Lage ihre Anforderungen an eine Softwarelösung vollständig zu definieren. Es erfolgt vielfach nur noch ein reagieren auf Trends am Markt. Somit ist ein äußerst flexibler iterativer Prozess von Nöten.
- Die Softwareerstellung ist ein sehr strukturierter Prozess, welcher somit ein geringeres Maß an Steuerung und Projektmanagement benötigt.[Ang05]

Somit werden schlanke und agile Projektmanagement Methoden benötigt um aktuelle Projekte umzusetzen. Durch die Nutzung von iterativen Methoden können später auftretende Anforderungen mit umgesetzt werden und die anfangs groben Anforderungen werden fortlaufend detailliert. Durch weniger Vorgaben und schlanke Managementprozesse wird insgesamt weniger Overhead an Managementtätigkeiten erzeugt, wodurch ein effizienteres Arbeiten möglich ist.

Dies bestätigt auch die folgende Abbildung 8.1, welche die Erfolgsquote von klassischen und agilen Projekten gegenüberstellt. Die Abbildung stammt aus einer 2009 vorgestellten Studie der oose Innovative Informatik GmbH, welche den Einfluss der klassischen und agilen Techniken auf IT-Projekte untersucht.

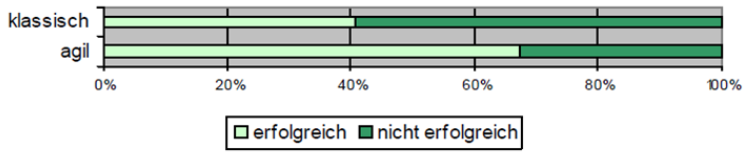


Abbildung 8.1: Erfolgsquote agiles / klassisches Projektmanagement [Inf09]

Folgend werden die Methoden des agilen Projektmanagements sowie die Methoden der agilen Softwareentwicklung beschrieben. Anschließend erfolgt die Erläuterung des daraus gewählten und auf unsere Bedürfnisse angepassten Vorgehensmodells.

8.1 Agiles Projektmanagement

Projektmanagement (PM) wird mittels verschiedenster Definitionen beschrieben. An dieser Stelle sei die Definition aus der DIN-Norm genannt, die einen möglichst allgemeinen Ansatz darstellt.

> *"Gesamtheit von Führungsaufgaben, -organisation, -techniken und -mitteln für die Initiierung, Definition, Planung, Steuerung und den Abschluss von Projekten."[DIN09]*

Agilität als Charakteristika des agilen Projektmanagements steht für eine Art der Beweglichkeit des Managements, um auf Änderungen einzugehen. Es basiert auf dem agilen Manifest aus der Softwareentwicklung. Dieses wurde im Jahr 2001 formuliert und umfasst folgende Bestandteile:

1. Individuen und Interaktionen gelten mehr als Prozesse und Tools.
2. Funktionierende Programme gelten mehr als ausführliche Dokumentation.
3. Die stetige Zusammenarbeit mit dem Kunden steht über Verträgen.
4. Der Mut und die Offenheit für Änderungen stehen über dem Befolgen eines festgelegten Plans. [al01]

Die Kombination aus diesen beiden Ansätzen stellt das Agile Projektmanagement dar. Es wird dabei eher als Werkzeugkasten verschiedener Methodiken angesehen, welche man nutzen kann, allerdings nicht muss. Folgend werden noch einige grundsätzliche Begriffe des Projektmanagements definiert, die sich in allen agilen Methoden wiederfinden. Die Definitionen werden dafür aus dem Glossar des Oose Engineering Process genommen.

Iteration *„Eine Iteration ist ein in ähnlicher Weise mehrfach vorkommender Zeitabschnitt in einem Prozess. Iterationen können Timeboxen sein. Eine Iteration entsteht durch die zeitliche Einteilung des Entwicklungsprozesses in mehrere gleichartige Zeitabschnitte (Iterationen). Innerhalb jeder Iteration wird gewöhnlich ein Mikroprozess durchlaufen (Analyse - Design - Implementierung - Test).“ [Inf10]*

Release *„Ein Build, das an den Auftraggeber (bzw. Produktempfänger) ausgeliefert werden kann. Ein Build ist eine gewöhnlich unvollständige und vorübergehende, aber ausführbare Version eines in Entwicklung befindlichen Systems.“ [Inf10]*

Meilenstein *„Ein Meilenstein definiert einen Termin, zu dem eine Menge von Ergebnissen in vorgegebener Detaillierung und Vollständigkeit nachprüfbar und formal dokumentiert vorliegen muss. Liegen die Ergebnisse zum geplanten Termin nicht vor, wird der Meilenstein verschoben. Ein Meilenstein ist ein Hilfsmittel zur Planung und Überwachung eines Entwicklungsprozesses.“ [Inf10]*

Ablauf in agilen Projekten

In agilen Projekten wird am Anfang noch keine genaue Definition des Zieles benötigt. Das Ziel befindet sich, abstrakt gesehen, in einer Wolke aus möglichen Lösungen. Man nähert sich diesem Ziel in mehreren Iterationen und kann so auf Veränderungen oder neue Anforderungen eingehen. Die folgende Abbildung 8.2 beschreibt den iterativen Ablauf nochmals genauer. Zusätzlich zu den Iterationen sollten Releases definiert werden. Die Releases enthalten die umzusetzenden Ziele und geben unter anderem so einen Zeitrahmen für das Projekt. [OB08]

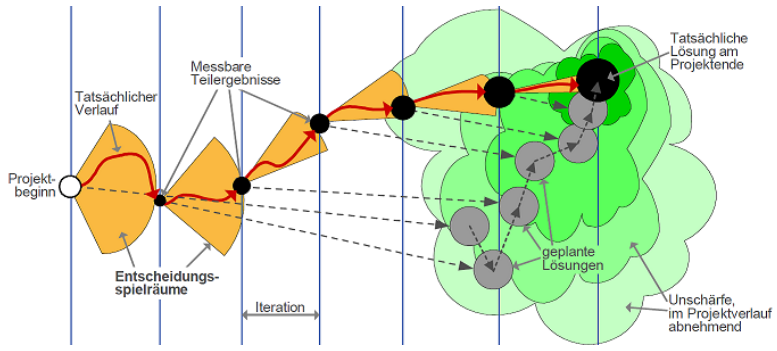


Abbildung 8.2: Ablauf agiles Projekt [OB08]

Unterschiede zum klassischen Projektmanagement

Folgend sollen die Unterschiede zwischen klassischem und agilem Projektmanagement anhand von fünf Beispielen dargestellt werden. Die Beispiele sollen die Vorteile vom agilen Projektmanagement verdeutlichen und stellen keine gesamte Betrachtung des Themas dar. [S.09]

Stabilität vs. Dynamik

Klassische Projekte gelten aufgrund der anfangs bereits definierten Anforderungen bereits als sehr stabil, da keine großen Änderungen mehr zu erwarten sind. Bei agilen Projekten hingegen werden anfangs nur grobe Anforderungen definiert, welche im Projektverlauf detaillierter werden. Somit ist eine gewisse Dynamik gegeben. Zudem wird versucht den Management-Overhead in agilen Projekten zu minimieren.

Kompliziert vs. Komplex

Für komplexe Projekte eignen sich agile Methoden deutlich besser, da anfangs vielfach noch keine Lösungsstrategie vorhanden ist. Diese kann über die Zeit gebildet werden. Zudem kann, durch kurze Iterationszyklen, auf Änderungen der Umwelt eingegangen werden. Metaphorisch ist die Kompliziertheit mit einem Schachspiel zu vergleichen, welches eine gewisse Menge an Zügen ermöglicht. Komplexität hingegen wäre ein Schachspiel, bei dem sich in gewissen Abständen die Regeln ändern würden. So muss man sich dauerhaft auf neue Situationen einstellen.

Top Down vs. Bottom Up

Im klassischen Projektmanagement wird vielfach eine Planung Top Down durch das Projektmanagement vorgenommen. Dies trifft auf das agile PM nicht zu. Es erfolgt die Planung des Projektes durch das gesamte Team und das Projektmanagement wirkt moderierend.

Delegativ vs. Partizipativ

Beim Agilen PM werden alle Projektteilnehmer in die Entscheidungsprozesse eingebunden und es wird jeweils ein Team-Agreement eingeholt. Dieser sehr partizipative Führungsstil erfordert einerseits viel Engagement vom Projektteam ist aber andererseits sehr motivierend. Beim klassischen PM erfolgt das Management viel delegativer durch die Projektleitung.

Verträge vs. Vertrauen

Im klassischen PM werden mit dem Kunden sämtliche Leistungen über Verträge geregelt, die den genauen Umfang des Projektes und die Anforderungen an das Produkt darstellen. Das agile PM setzt dabei mehr auf ein vertrauensvolles Verhältnis zwischen dem Kunden und dem Projektteam.

8.2 Agile Softwareentwicklung

> *„Agility is the ability to both create and respond to change in order to profit in a turbulent business enviroment.“* [HHE02]

Agile Softwareentwicklung kann als eine Gegenbewegung zu den klassischen Methoden und Vorgehensmodellen der Softwareentwicklung, wie zum Beispiel dem Wasserfallmodell, gesehen werden [Fow01, HHI07]. Anstatt einem strikten Plan zu folgen, soll durch den Einsatz agiler Methoden flexibel auf jede Art von Änderungen reagiert werden können [BBVB+01b, Fow01]. Die agile Softwareentwicklung ist kein konkretes Vorgehensmodell sondern eher eine Art Philosophie, auf der dann konkrete agile Prozesse, wie zum Beispiel Extreme Programming, Scrum oder Crystal basieren [Fow01]. Im Folgenden werden die allgemeinen Ideen, die hinter der agilen Softwareentwicklung stehen, näher beschrieben. Dies geschieht anhand des, in Abschnitt 8.1 bereits erwähnten, agilen Manifests.

8.2.1 Agile Manifesto

Ein Versuch zu beschreiben, was die agile Softwareentwicklung ausmacht, wurde im Jahr 2001 mit dem *Agile Manifesto* [BBVB+01b] unternommen. Das Manifest wurde von 17 Vertretern verschiedener agiler Methoden, welche die agile Softwareentwicklung maßgeblich beeinflusst haben, ausgearbeitet und unterzeichnet.

> *„We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> - ***Individuals and interactions*** *over processes and tools*
> - ***Working software*** *over comprehensive documentation*
> - ***Customer collaboration*** *over contract negotiation*
> - ***Responding to change*** *over following a plan*

> *That is, while there is value in the items on the right, we value the items on the left more.“*

Die vier zentralen Punkte des *Agile Manifesto* werden im Folgenden noch einmal näher erläutert.

Individuals and interactions over processes and tools

> Software wird von Menschen entwickelt, die jeder eine eigene Persönlichkeit haben. Bei den klassischen Vorgehensmodellen der Softwareentwicklung wird in der Regel so vom Individuum abstrahiert, dass der Entwicklungsprozess von beliebigen Personen durchführbar ist [Rog09]. Das *Agile Manifesto* weicht von dieser Sichtweise ab und stellt die Individualität der einzelnen Personen in den Fokus der Softwareentwicklung. Individualität ist insoweit eine wichtiger Faktor bei der Entwicklung von Software, als Softwareentwicklung ein kreativer Prozess ist [Rog09]. Die Höhergewichtung von Interaktion über den Einsatz von Tools zielt auf die zwischenmenschliche Kommunikation ab. Spezielle Werkzeuge können diese zwar unterstützen, jedoch nicht ersetzen [Rog09].

Working software over comprehensive documentation

> Lauffähige Software ist nach den agilen Prinzipien [BBVB+01a] das Maß für den Fortschritt der Entwicklung und wird dementsprechend höher gewertet als Dokumentation. Dokumentation meint hierbei nicht die Dokumentation der Software, sondern die großen, bei klassischen Vorgehensmodellen anfallenden, Dokumente, wie zum Beispiel eine im Vorfeld der Entwicklung erstellte, umfassende Anforderungsanalyse [Rog09]. Bei der agilen Softwareentwicklung soll sich hingegen auf die Auslieferung lauffähiger Software konzentriert, und nur das dokumentiert werden, was als unbedingt notwendig erachtet wird [FH01].

Customer collaboration over contract negotiation

> Verträge mögen zwar eine gute Grundlage für die Entwicklung von Software sein, reichen in der Regel jedoch nicht aus, um die Software zu entwickeln, die der Kunde auch tatsächlich benötigt [FH01]. Um Missverständnissen und Unstimmigkeiten zwischen Kunden und Entwicklern vorzubeugen und die realen Anforderungen der Software zu ermitteln, ist eine gute Zusammenarbeit und Kommunikation dieser beiden Parteien unabdingbar. Im Endeffekt verfolgen Kunden und Entwickler das gleiche Ziel, nämlich die Entstehung einer vernünftigen Software, und sollten deshalb auch an einem Strang ziehen.

Responding to change over following a plan

> Im Umfeld von Softwareprojekten kommt es zu ständigen Veränderungen. Diese können wirtschaftlicher Natur, wie zum Beispiel eine Wirtschaftskrise, oder

> technischer Natur, wie zum Beispiel die Einführung eines neuen Standards, sein, sowie aus unklaren oder sich ändernden Anforderungen resultieren. Die Methoden der agilen Softwareentwicklung „müssen in der Lage sein, mit beliebigen Änderungen umzugehen, also insbesondere mit solchen, die man im Vorfeld nicht antizipiert hat“ [Rog09]. Planung ist zwar auch ein Bestandteil der agilen Softwareentwicklung, jedoch sollte ein Plan nicht starr verfolgt werden, sondern flexibel und anpassbar sein. Aus Veränderungen lassen sich, durch die Flexibilität agiler Prozesse, dann sogar Wettbewerbsvorteile für den Kunden generieren [BBVB+01a].

Hinter dem *Agile Manifesto* an sich stehen 12 Prinzipien (siehe [BBVB+01a]), die im Folgenden zusammengefasst dargestellt werden. Agile Prozesse fördern die Kundenzufriedenheit durch kurze Releasezyklen und sind offen für Feedback und Veränderungen. Dafür sollten Kunden und Entwickler möglichst eng zusammenarbeiten. Der bevorzugte Kommunikationsweg stellt dabei die Kommunikation von Angesicht zu Angesicht dar, was im Übrigen für alle Bereiche der agilen Softwareentwicklung gilt. Projektmitarbeiter sollten in einer möglichst motivierenden und vertrauensvollen Umgebung arbeiten. Die Entwicklungsteams sollten sich dabei soweit wie möglich selbst organisieren. In regelmäßigen Abständen werden die verwendeten agilen Methoden reflektiert und gegebenenfalls angepasst oder verändert. Insgesamt gesehen sollten die verwendeten agilen Methoden möglichst einfach gehalten werden. Dies gilt für Managementmethoden genauso wie für Entwicklungsmethoden.

8.3 Beschreibung Vorgehensmodell

Das gewählte Vorgehensmodell beinhaltet Bestandteile des Oose Engineering Process, des Scrum sowie des Extrem Programming. Es wurde, wie in agilen Projekten üblich, nicht ein vorgegebenes Vorgehensmodell in vollem Umfang umgesetzt, sondern ein adaptiertes Vorgehensmodell verwendet. Die grundlegenden drei Modelle werden folgend kurz erläutert.

8.3.1 Oose Engineering Process

Der Oose Engineering Process (OEP) wurde von der Firma oose Innovative Informatik GmbH entwickelt und gilt als umgreifende Methode des agilen Projektmanagements. Es existiert eine sehr ausführliche Onlinedokumentation sowie Schulungsunterlagen zum OEP. Bernd Oestereich und Christian Weiss definieren den Prozess folgendermaßen:

> *„Der OEP ist ein Vorgehensmodell und Leitfaden zur agilen objektorientierten Softwareentwicklung, basierend auf UML und Unified Process und einer Abbildungskonvention auf das V-Modell XT.“ [OB08, S. 423]*

Aus dieser Definition ist zu erkennen, dass der OEP versucht das klassische Projektmanagement und das agile Projektmanagement, wie es unter anderem in Scrum praktiziert wird, zu verknüpfen.

Auf oberster Ebene ist der OEP in Phasen und Disziplinen aufgeteilt. Die Phasen stellen dabei eine zeitliche Gliederung des Entwicklungsprozesses dar. Jede Phase hat dabei einen Zweck und definierte Ergebnisse, welche mit einem Meilenstein geprüft werden. In der folgenden Grafik stellen die Phasen die horizontale Achse dar. Jede Phase wird dabei durch mindestens eine Iteration durchgeführt. So bleibt die Agilität im Projekt erhalten. Die Disziplinen entsprechen einer inhaltlichen Gliederung aller Aktivitäten. Es wird dabei jedoch nochmals unterschieden zwischen Kerndisziplinen, welche den Systemerstellungsprozess unterstützen, und unterstützenden Disziplinen, welche nur indirekt zur Systemerstellung beitragen. Aktivitäten sind dabei konkrete Arbeitsaufgaben, die von einer verantwortlichen Person übernommen werden und klare Ergebnisse sowie Zwischen- oder Endzustände haben. Die Granularität der Aufgaben sollte dabei den vorher genannten Punkten angepasst werden, so dass z.B. eine Person eine Arbeitsaufgabe auch in einer Iteration bewältigen kann. [IIG07]

Die folgende Abbildung 8.3 zeigt den Zusammenhang zwischen Phasen und Disziplinen nochmals deutlicher und stellt auch schematisch die zu erwartende Arbeitsbelastung des Teams dar. Die Arbeitsbelastung errechnet sich aus der Menge aller Aktivitäten die in einer Disziplin und in einer Phase durchgeführt werden müssen. Die Meilensteine am Ende einer jeweiligen Phase sind mit den roten Dreiecken dargestellt.

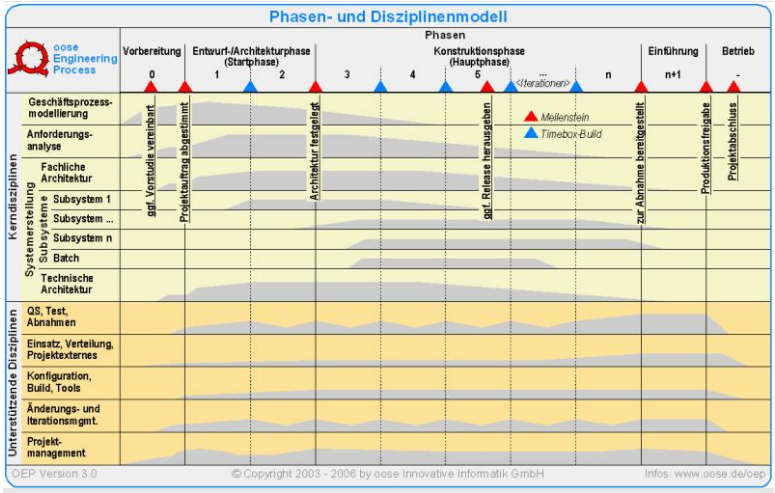


Abbildung 8.3: Aufbau OEP[IIG07]

8.3.2 Scrum

Die Begründer Jeff Sutherland und Ken Schwaber nannten das von ihnen entwickelte Vorgehensmodell Scrum (deutsch: Gedränge) in Anlehnung an eine frühere Ausarbeitung von Nonaka und Takeuchi. Scrum soll dabei für den Zusammenhalt im Projektteam stehen und für die Nutzung weniger Regeln. [B.09, S. 8ff]

Im Scrum werden nur drei Rollen (Product Owner, Scrum Master, Team) definiert. So ist eine sehr flache Hierarchie möglich, was auch ein Zeichen für einen relativ kleinen Management- und Koordinationsaufwand ist. Scrum unterscheidet zudem sehr strikt zwischen Projektbeteiligten, welche in der Rollenstruktur abgebildet werden, und Personen, die nur am Projekt interessiert sind aber daran nicht arbeiten. Die Anforderungen und die Umsetzung der Anforderungen werden in Artefakten festgehalten, welche folgend kurz erläutert werden.

Product Backlog

Der Product Backlog beschreibt die Anforderungen an das Produkt. Jedoch werden diese Anforderungen nicht im klassischen Sinne aufgenommen, sondern ergeben sich mehr aus Eigenschaften und Merkmalen des Produktes. Aus dem Product Backlog werden die priorisierten Anforderungen des Projektes in die jeweiligen Sprints übertragen und abgearbeitet. Da Scrum ein agiler Prozess ist, ist der Product Backlog nie vollständig. [B.09, S. 122ff]

Sprint Backlog

Der Sprintbacklog setzt sich aus dem Product Backlog zusammen und stellt die Anforderungen dar, welche in dem aktuellen Sprint umgesetzt werden sollen. Diese Anforderungen werden folgend in kleinere Arbeitspakete aufgeteilt. Dabei sollte die Größe eines Arbeitspaketes einen Bearbeitungsrahmen von 4-16St. nicht verlassen. Zudem können Verantwortlichkeiten, Status und bereits erfolgte Aufwände dargestellt werden. So informiert der Sprint Backlog sehr genau in welchem Schritt sich das Team gerade befindet und wie groß die Fertigstellungswahrscheinlichkeit ist. [K.04, S. 12ff]

Abschließend werden noch die Abläufe im Scrum dargestellt, welche sich über die Meetings definieren. Die Meetings bilden dabei einen zentralen Bestandteil im Scrum-Prozess, da im Scrum ein demokratischer Führungsstil angewendet wird.

Sprintplanung

Das Sprintplanungsmeeting ist aufgeteilt in zwei Bestandteile. Im ersten Teil beschreibt der Product Owner die am höchsten priorisierten Anforderungen aus dem Product Backlog. Anschließend wählt das Team alle Anforderungen aus, die es meint im folgenden Sprint umsetzen zu können. Im zweiten folgenden Teil des Meetings plant das Team anschließend den genauen Ablauf des Sprints und definiert Arbeitsaufgaben für die einzelnen Teammitglieder. Das gesamte Meeting soll maximal acht Stunden dauern. [K.04, S.8]

Sprint Review

Beim SprintReview werden die erarbeiteten Ergebnisse dem Product Owner und allen Anteilseignern (wenn interessiert) präsentiert. Es erfolgt somit eine Abnahme der Arbeit aus dem Sprint. Zudem kann eine Reflektion des letzten Sprints angeschlossen werden, in der eventuelle Probleme und Unstimmigkeiten geklärt werden. Zudem erfolgt eine Feedbackrunde über den eigentlichen Scrumprozess. [K.04, S.8]

Daily Scrum Meeting

Das Daily Scrum Meeting findet täglich statt und soll über aktuelle Lage im Team informieren. Dabei ist es meist stark normiert, hat eine definierte Länge von 15 Minuten und jedes Teammitglied beantwortet definierte Fragen. Diese können wie folgt sein:

- Welche Arbeitsaufträge wurden abgeschlossen?
- Welche Arbeitsaufträge sind neu begonnen worden?
- Was steht mir im Weg, um meine Arbeitsaufträge zu erledigen?
- Welche Arbeitsaufträge sind in Arbeit?
- Was will ich bis zum nächsten Treffen erreichen? [OB08, S. 183]

8.3.3 Extreme Programming

Extreme Programming (XP) ist ein konkretes Vorgehensmodell der agilen Softwareentwicklung. Die Entwicklung von XP wird hauptsächlich Kent Beck zugeschrieben und geht auf das Chrysler Comprehensive Compensation System (C3) Projekt zurück, in dem Beck ab 1996 als Projektleiter tätig war [Rog09]. Im Jahr 1999 veröffentlichte Beck das Buch „Extreme programming explained: embrace change“ [Bec99], welches erstmals

das Vorgehensmodell von XP beschrieb. Wie ein Großteil der verfügbaren Literatur, bezieht sich die nachfolgende Betrachtung von XP auf diese erste Beschreibung Becks [Fow01]. In XP geht es um die Anwendung verschiedener Programmiertechniken, klare Kommunikation und Teamwork [BA04]. Neben technischen Aspekten spielen auch aber soziale Faktoren eine große Rolle.

Werte

XP definiert vier Werte, die den Rahmen der Softwareentwicklung vorgeben. Diese Werte sind Kommunikation, Einfachheit, Feedback sowie Mut und werden im Folgenden näher beschrieben.

Kommunikation

> Kommunikation ist ein essentieller Bestandteil der Softwareentwicklung im Allgemeinen und von XP im Speziellen. Viele Probleme, die in Softwareprojekten auftreten sind das Resultat mangelnder Kommunikation [Bec99]. So kommt es beispielsweise zu Fehlern, weil gewichtige Designänderungen nicht kommuniziert wurden oder die Anforderungen werden nicht vollständig ermittelt, da die Kommunikation mit dem Kunden nicht richtig funktioniert. XP setzt sich für eine klare und ehrliche Kommunikation zwischen allen Projektbeteiligten ein und bietet eine Reihe von Praktiken, um diese Idee zu verwirklichen [Bec99]. Direkte, zwischenmenschliche Kommunikation wird von XP als der effektivste Kommunikationsweg angesehen, wobei auch Programmierartefakte, wie Kommentare oder die Namensgebung, einen Teil zur Kommunikation beitragen können [Bec99, Mar05].

Einfachheit

> „Was ist die einfachste Lösung die funktionieren könnte?“ Die Frage beschreibt die Idee der Einfachheit in XP [Bec99]. Auch wenn dieses Prinzip, im wahrsten Sinne des Wortes, einfach erscheint, so stellt es häufig eine große Herausforderung für die Entwickler dar. Während man geneigt ist Dinge, die in der nahen Zukunft benötigt werden, schon in der aktuellen Lösung zu berücksichtigen, beispielsweise durch generische Schnittstellen, soll sich in XP nur auf das aktuell zu lösende Problem konzentriert und jegliche Komplexität dabei entfernt werden [Bec99]. XP geht davon aus, dass es besser ist, heute eine Sache einfach zu lösen und später gegebenenfalls etwas mehr zu investieren, falls diese Lösung angepasst werden muss, als von vornherein eine komplexere Lösung zu entwickeln, die nie benötigt wird. XP trifft dabei die Annahme, dass durch die angewandten Praktiken, die Kosten für Änderungen nicht exponentiell wachsen [Bec99].

Feedback

Feedback ist eine Größe, die den Prozess der Softwareentwicklung beeinflusst. Feedback kann aus verschiedenen Quellen stammen, wobei eine der wichtigsten Feedback durch eine enge Zusammenarbeit mit dem Kunden ist [Mar05]. Feedback durch den Kunden führt in der Regel zu Änderungen, was von XP, als einen agilen Prozess, jedoch nicht als Bedrohung, sondern als Chance verstanden wird.

Mut

Neue Ideen auszuprobieren, unbequeme Entscheidungen zu treffen, offen miteinander umzugehen, all dies erfordert Mut [Bec99]. Mut bedeutet jedoch nicht, diese Dinge ohne Rücksicht auf Verluste durchzuführen. Deshalb ist Mut ein Wert, der sich nur in Verbindung mit den drei anderen Werten sinnvoll in XP integriert [BA04]. In Kombination mit diesen Werten erlaubt Mut, auch schwierige Herausforderungen, wie zum Beispiel ein systemumfassendes Refactoring, anzugehen und zu meistern [Mar05].

Praktiken

Um die beschriebenen vier Werte in der Praxis umzusetzen, bietet XP verschiedene Praktiken. Die einzelnen Praktiken an sich wurden bereits vor der Entwicklung von XP erfolgreich eingesetzt und im Rahmen von XP, auf Basis der vier Werte, miteinander kombiniert. Das Ergebnis ist dabei mehr als die Summe der einzelnen Teile. Die einzelnen Techniken profitieren stark von Synergieeffekten untereinander [Bec99, Mar05, Fow01]. Im Folgenden werden einzelne XP Praktiken beschrieben, die im Rahmen der Projektgruppe eingesetzt wurden.

Kurze Releasezyklen

Kurze Releasezyklen ermöglichen zeitnahes Feedback und erlauben es, schnell und flexibel auf veränderte Anforderungen zu reagieren. XP setzt auf ein möglichst frühes Produktivsystem, dass eine Teilfunktionalität beinhaltet, die für den Kunden von Wert ist. Weitere Releases sollen in möglichst kurzen Abständen folgen und diejenigen Anforderungen umsetzen, die vom Kunden als am wichtigsten eingestuft wurden [Bec99]. So kann der Kunde regelmäßig Feedback geben und den Projektverlauf steuern, indem er beispielsweise die Prioritäten der Anforderungen ändert.

Ständig verfügbarer Kunde

Zum Entwicklungsteam sollte nach Möglichkeit ein realer Kunde, das heißt ein tatsächlicher Benutzer des Systems, gehören. Dieser ist dafür zuständig Fragen zu

beantworten, Probleme zu klären und feingranulare Prioritäten zu setzen [Bec99]. In der Praxis lässt sich die physische Anwesenheit des Kunden häufig nicht realisieren. In diesem Fall sollte der Kunde jedoch möglichst jederzeit erreichbar sein, so dass eventueller Klärungsbedarf nicht die Arbeit des ganzen Teams blockiert.

Pair Programming

Pair Programming ist eine Methode, bei der Software von zwei Programmierern gemeinsam an einem Rechner entwickelt wird. Während der eine Programmierer den eigentlichen Code schreibt, analysiert der andere Programmierer den geschriebenen Code und denkt auf einer eher strategischen Ebene. Dadurch entsteht beim Programmieren ein Dialog darüber, wie das aktuelle Problem oder die aktuelle Aufgabe am besten gelöst werden kann [Bec99, BA04].

Beim Pair Programming sollen die Rollen innerhalb der Paare regelmäßig gewechselt werden. Das gleiche gilt für die Paarungen an sich. Unter welchen Bedingungen sich die Paare zusammenfinden ist dem Team selbst überlassen, so können sich beispielsweise Paare bilden, die sich in ihren Kompetenzen bei der Lösung eines bestimmten Problems ergänzen, aber auch das zufällige Bilden von Paaren ist eine Möglichkeit. Möchten zwei Teammitglieder aus irgendwelchen Gründen nicht zusammen programmieren, so ist dies zu akzeptieren, um nicht noch mehr Unstimmigkeiten im Team hervorzurufen [Bec99].

Dadurch das simultan zur Programmierung ein Code Review stattfindet und immer ein Programmierer strategisch mitdenkt, können Fehler bereits im Vorfeld vermieden und eine allgemeine Steigerung der Code- und Designqualität erreicht werden [Bec99]. Weiterhin fördert Pair Programming die Diffusion von Wissen innerhalb des Teams, indem die Paarungen regelmäßig wechseln und die Entwickler ihre Erfahrungen untereinander austauschen [Bec99, Rog09]. Auch die disziplinierte Anwendungen anderer XP Praktiken kann durch Pair Programming unterstützt werden. Der zweite Programmierer übernimmt hierbei die Rolle des „schlechten Gewissens“, das verhindert, dass Praktiken, wie zum Beispiel das Refactoring, vernachlässigt werden. Dieses „schlechte Gewissen“ bewirkt außerdem, dass sich die Entwickler auf ihre Arbeit konzentrieren, anstatt Nebenbeschäftigungen, wie beispielsweise dem Abrufen von privaten Emails, nachzugehen [Bec99, Rog09]. All diese erzielten Vorteile entkräften zudem die immer wieder angeführte Behauptung, dass sich durch Pair Programming die Produktivität der Softwareentwickler halbiert. Dazu sei außerdem noch erwähnt, dass sich die Produktivität eines Softwareentwicklers nicht durch seine Tippgeschwindigkeit bestimmt und somit eine Halbierung der Tippgeschwindigkeit nicht zu einer Halbierung der Produktivität führt [Rog09].

Einfaches Design

Die Praktik des einfachen Designs bezieht sich direkt auf die Einfachheit, als Einer der vier Werte des XP. Im Gegensatz zum Systementwurf vor der Implementierung, wie es bei den klassischen Vorgehensmodellen der Softwareentwicklung üblich ist, entwickelt und verbessert sich das Design bei XP im Laufe der Implementierung [BA04]. Dabei soll stets das einfachste Design entstehen, welches zur Erreichung der aktuellen Ziele benötigt wird. Dieses optimale Design definiert sich nach Beck [Bec99] dadurch, dass es alle Test besteht, keine duplizierte Programmlogik beinhaltet, den Inhalt des Systems verständlich ausdrückt und die kleinstmögliche Anzahl an Klassen und Methoden besitzt. Mit dem Refactoring bietet XP eine Praktik, welche die ständigen Designänderungen und somit einfaches Design ermöglicht.

Refactoring

Refactoring beschreibt den Vorgang der Restrukturierung von Softwarecode ohne dabei die Funktionalität des Systems zu beeinflussen. Refactoring unterstützt einfaches Design und wird immer unter der Fragestellung durchgeführt, wie das aktuelle Design einfacher oder verständlicher gemacht werden kann [Bec99]. Ein typischer und einfacher Anwendungsfall ist beispielsweise das Entfernen von duplizierter Programmlogik. Bei größeren Umstrukturierungen des Systems sollte beim Refactoring iterativ, das heißt in kleinen Schritten, vorgegangen werden [Bec99]. Ständiges Refactoring ist ein Mittel um stets ein einfaches und leicht verständliches System zu erhalten, in dem Änderungen ohne großen Aufwand, das bedeutet auch ohne große Kosten, durchgeführt werden können.

Gemeinsame Verantwortlichkeit

Gemeinsame Verantwortlichkeit bedeutet, dass das gesamte Team die Verantwortung für das gesamte System übernimmt. Bei gemeinsamer Verantwortlichkeit gibt es keine Verantwortlichen für spezielle Module oder Teile des Codes. Dies hat zur Folge, dass jeder Entwickler jeden Teil des Codes verändern kann, sobald er eine Idee zur Verbesserung hat. Gemeinsame Verantwortlichkeit reduziert die Abhängigkeit des Teams von einzelnen Personen [Bec99].

Programmierstandards

Da jeder Entwickler jeden Teil des Codes ändern kann, man zusammen in Paaren programmiert und ständige Refactorings des Codes stattfinden, sind Programmierstandards oder Guidelines, wie zum Beispiel Guidelines zur Codeformatierung, für die Arbeit in XP-Teams notwendig [Bec99].

Soziale Aspekte

Ein weiteres wichtiges Merkmal von XP ist, dass XP versucht die Bedürfnisse der involvierten Menschen zu berücksichtigen. Jeder Mensch hat seine eigene Persönlichkeit, die es zu respektieren gilt. Eine große Herausforderung besteht darin, die Bedürfnisse der Einzelnen mit den Bedürfnissen des Teams ins Gleichgewicht zu bringen [BA04].

Ein Punkt, in dem sich der soziale Aspekt von XP manifestiert, ist die große Bedeutung der zwischenmenschlichen Kommunikation in XP. Bereits als einer der vier Werte beschrieben, setzen fast alle Praktiken in XP Kommunikation voraus. Um diese weiter zu fördern, schlägt Beck [Bec99] zum Beispiel das gemeinsame Arbeiten in einem Raum vor. Des Weiteren werden Entscheidungen, beispielsweise über den Einsatz bestimmter Praktiken, nicht von oben herab entschieden, sondern im Team diskutiert und abgestimmt.

XP geht davon aus, dass zufriedene und motivierte Teammitglieder produktiver arbeiten, als unzufriedene [BA04]. Eine angenehme Arbeitsatmosphäre, gute Beziehungen innerhalb des Teams, das Akzeptieren menschlicher Fehler sowie gegenseitiges Vertrauen helfen dabei, ein Umfeld zu schaffen, in dem Softwareentwickler ihrer Arbeit mit Freude nachgehen können.

8.4 Anwendung Vorgehensmodell

Für die Durchführung der Projektgruppe wurde keins der drei vorgestellten Vorgehensmodelle komplett übernommen. Stattdessen wurde ein adaptiertes eigenes Modell entwickelt, welches sich aus den vorgestellten Modellen zusammensetzt. Dabei wurde auf einen möglichst kleinen Overhead an Organisations- und Kontrollfunktionen Wert gelegt. Somit konnte die meiste Arbeitszeit auf produktive Themen fixiert werden, sodass ein rasches Fortkommen des Projekts ermöglicht wurde. Abbildung 8.4 bietet einen Überblick über unser Vorgehensmodell, welches im Folgenden näher erläutert wird.

8.4.1 Planung

Die Planungsstrategie unseres Modells setzt sich aus einer Release- und Iterationsplanung zusammen. Für die Releases werden jeweils Product-Backlogs angelegt, welche den Umfang der Realisierung beschreiben. Diese Product-Backlogs bestehen aus einer Liste von Anwendungsfällen, die ggf. noch weiter in spezielle Arbeitspakete untergliedert werden müssen. Die Liste ist ein „lebendes“ Dokument, das im Laufe des Projekts permanent gepflegt wird.

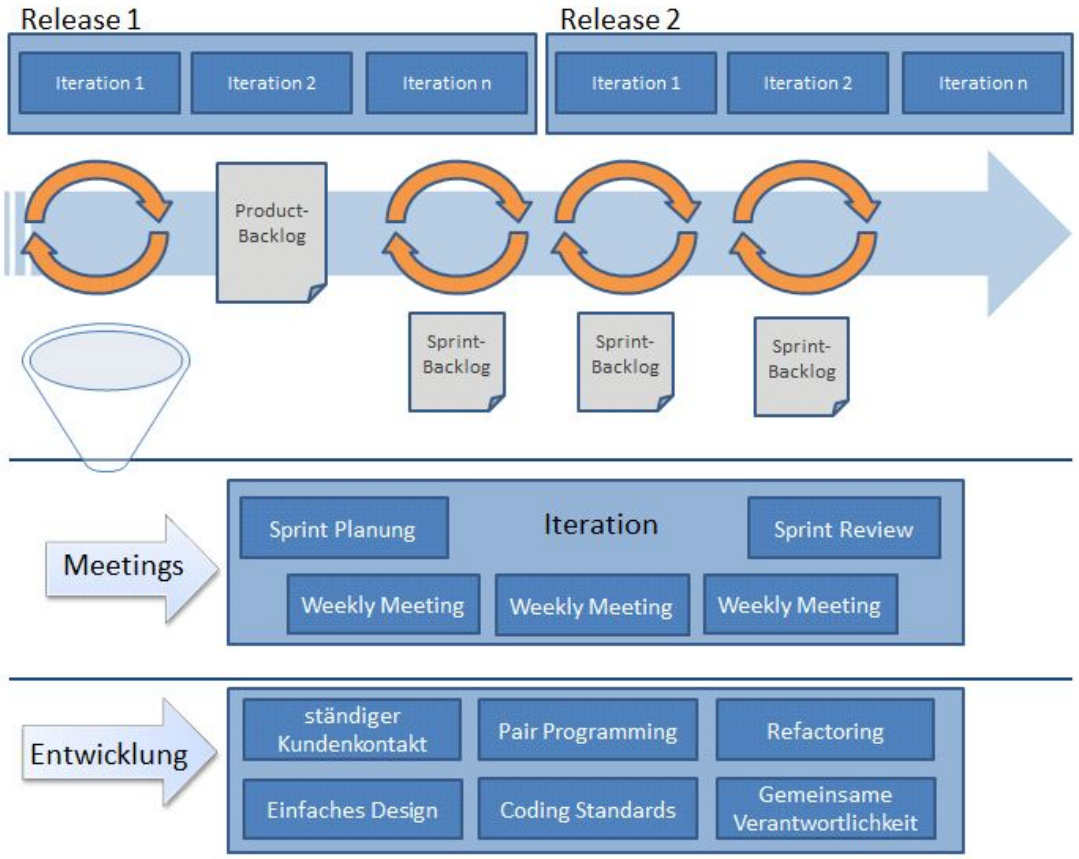


Abbildung 8.4: Projekt Vorgehensmodell

Die einzelnen Releases sind wiederum in Iterationen aufgeteilt, die einen Zeitraum von ca. vier Wochen umfassen. Für jede Iteration, auch Sprint genannt, wird ein Spring-Backlog angelegt. Dieses stellt einen Ausschnitt aus dem Product-Backlog dar und umfasst alle Anwendungsfälle, die in der jeweiligen Iteration zu bearbeiten sind. Die Anwendungsfälle für die Sprint-Backlogs werden durch das Team ausgewählt und vom Aufwand eingeschätzt. Sofern Arbeitspakete am Ende einer Iteration nicht abgeschlossen sind, muss entschieden werden ob man diese mit in die neue Iteration überführt oder die vorherige Iteration verlängert. Dies sollte in Abhängigkeit zur Menge entschieden werden.

8.4.2 Kommunikation

Über eine Meetingstruktur wird die Kommunikation mit dem gesamten Team sichergestellt. Die Meetings finden in der Regel wöchentlich statt und stellen eine Plattform für den Austausch zum aktuellen Fortschritt dar. Zu Iterationsbeginn wird

zudem die vorherige Iteration besprochen und die neue eingeführt. Dabei werden die umzusetzenden Arbeitspakete konkretisiert und im Stundenumfang abgeschätzt. Die Meetingkultur orientiert sich stark am Scrum-Prozess.

Neben den Meetings findet eine permanente direkte Kommunikation zwischen den einzelnen Teammitgliedern statt. Arbeitspakete werden von mehreren Teammitgliedern in kleinen Gruppen oder Paaren bearbeitet und das Team arbeitet zusammen an einem Ort, nach Möglichkeit sogar in einem Raum.

Durch diese Strukturen soll versucht werden großen Aufwand im Projektcontrolling und in der Projektdokumentation zu vermeiden. Zudem wird so der anvisierte demokratische Führungsstil unterstützt.

8.4.3 Entwicklung

Im Bereich der Softwareentwicklung werden verschiedene Praktiken aus dem Extreme Programming angewendet.

Der ständige Kontakt zum Kunden, in unserem Fall zu den Haupt-Steakholdern, ermöglicht einen Austausch über die Anforderungen zwischen Kunden und Entwicklern und bringt dem Team kontinuierliches Feedback. Durch den Einsatz von Pair Programming soll zum einen die Codequalität erhöht werden und zum anderen ein Wissenstransfer stattfinden. Einzelne Arbeitspakete werden von Paaren bearbeitet. Die Zusammensetzung der Paare kann dabei im Laufe einer Iteration mehrfach wechseln, was den Wissenstransfer fördert. Paarungen von erfahrenen Entwicklern mit weniger erfahreneren Entwicklern sollen helfen schnelle Lerneffekte zu erzielen. Durch einfaches Design, ständige Refactorings und Programmierstandards soll der Softwarecode und dessen Struktur stets einfach und für alle Teammitglieder verständlich gehalten werden. Dies unterstützt die iterative Vorgehensweise, indem Änderungen jederzeit und ohne exponentielle Aufwandssteigerung durchgeführt werden können. Im Endeffekt sollte sich jedes Teammitglied für das gesamte System verantwortlich fühlen. Durch gemeinsame Verantwortlichkeit reduziert sich die Abhängigkeit des Teams von einzelnen Personen, sodass zum Beispiel Ausfälle durch Krankheit leichter kompensiert werden können.

Kapitel 9

Umsetzung Vorgehensmodell Projektmanagement

Für die Umsetzung des im vorherigen Kapitel vorgestellten Vorgehensmodells wurden diverse Werkzeuge eingesetzt, die im folgenden Kapitel erläutert werden. Diese Werkzeuge, insbesondere das Trac-System, bildeten die Kommunikationsgrundlage für das Projektmanagement.

Die Meetings sowie der Entwicklungsprozess wurden anhand der Modellstruktur ausgerichtet. Somit folgt der Projektablauf, wo möglich, dem Vorgehensmodell. Der Projektablauf wird folgend auch nochmal genauer erläutert. So kann die Arbeitsweise über das gesamte Projekt betrachtet werden.

Ein weiterer wichtiger Punkt bei der Projektdurchführung war die Rollenstruktur im Projektteam. Diese wird in dem folgenden Kapitel aufgegriffen und genauer erläutert. Durch die Rollenstruktur waren die Zuständigkeiten in der Projektgruppe klar gegliedert, wodurch konkrete Ansprechpartner entstanden sind.

9.1 Werkzeuge

Während der Durchführung und Organisation der Projektgruppe wurden vom Projektmanagement und der Gruppe einige Hilfsmittel benutzt. Eine Auswahl dieser wird hier nun vorgestellt.

9.1.1 Trac mit agile42

Das zentrale Tool für die Verwaltung aller Aufgaben und Tätigkeiten war bei unserem Projekt das Trac. Trac ist ein webbasiertes System, welches dem Projektmanagement von Softwarelösungen dient.

Es bietet im Grundsystem ein Ticketing-System, ein Wiki-Tool sowie einen Zugriff auf das Subversion-Repository. Somit können in einer Timeline alle Änderungen im Repository nachverfolgt werden.

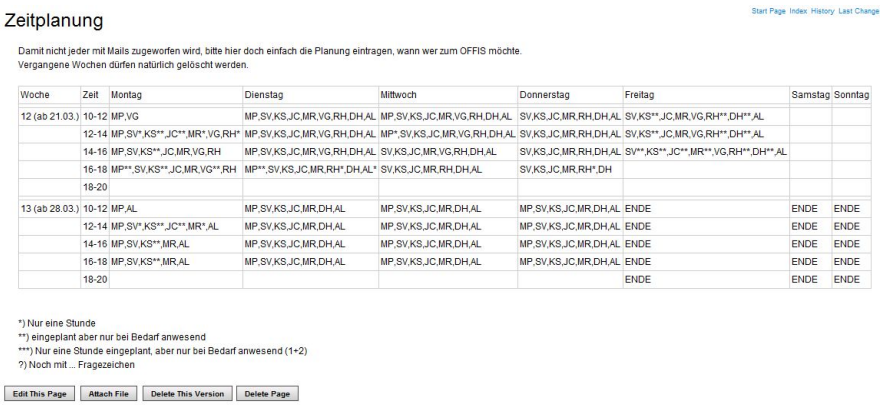
Für die Nutzung in der Projektgruppe wurde das Trac mit einer freien Version des Plugins agile42 erweitert. Agile42 ist eine Entwicklung der Agilo Software GmBW (`http://www.agilosofteware.com`) und gibt es in einer Pro sowie in einer frei erhältlichen Version. Mit diesem Plugin ist es möglich agile Methoden für das Projektmanagement zu nutzen. Es werden beispielsweise das Product Backlog sowie die Iterationen eingeführt. Zudem ist mit dem Plugin ein Projektcontrolling mittels Burndown-Charts möglich. Folgend werden die Funktionen nochmal kurz zusammengefasst:

- Konfigurierbares Product und Sprint Backlog
- Anpassbare Productbacklog
- Erstellen von Anforderungen und User Storys
- Priorisierung durch die Vergabe von Business Value Points
- Admin-Schnittstelle für das Hinzufügen, Ändern oder Konfigurieren von Items, Eigenschaften und Feldern
- Bi-direktionale Rückverfolgbarkeit, um Items zu verknüpfen und so das Produktmanagement zu erleichtern
- Real time Burn Down Chart, um jederzeit den Sprint Fortschritt ersehen zu können.
- Unterstützung von Multiple Teams und Sprints, um die Zusammenarbeit auch bei verteilten Teams zu erleichtern.

Folgend wird die Nutzung in unserem Projektkontext weiter erläutert und anhand von Abbildungen visualisiert.

Wiki

Das Wiki war zentraler Bestandteil für die Planung der Anwesenheitszeiten der Projektmitglieder in den Projekträumen. Zudem wurden informative bzw. wissenswerte Informationen dort abgelegt. Somit konnte vielfach durch einen Blick ins Wiki viel Arbeit und Zeit gespart werden. Folgende Abbildung 9.1 zeigt die Zeitplanung mit den jeweiligen Kürzeln der Projektmitgliedern.



Zeitplanung

Start Page Index History Last Change

Damit nicht jeder mit Mails zugeworfen wird, bitte hier doch einfach die Planung eintragen, wann wer zum OFFIS möchte.
Vergangene Wochen dürfen natürlich gelöscht werden.

Woche	Zeit	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
12 (ab 21.03.)	10-12	MP,VG	MP,SV,KS,JC,MR,VG,RH,DH,AL	MP,SV,KS,JC,MR,VG,RH,DH,AL	SV,KS,JC,MR,RH,DH,AL	SV,KS**,JC,MR,VG,RH**,DH**,AL		
	12-14	MP,SV*,KS**,JC**,MR*,VG,RH*	MP,SV,KS,JC,MR,VG,RH,DH,AL	MP*,SV,KS,JC,MR,VG,RH,DH,AL	SV,KS,JC,MR,RH,DH,AL	SV,KS**,JC,MR,VG,RH**,DH**,AL		
	14-16	MP,SV,KS**,JC,MR,VG,RH	MP,SV,KS,JC,MR,VG,RH,DH,AL	SV,KS,JC,MR,VG,RH,DH,AL	SV,KS,JC,MR,RH,DH,AL	SV**,KS**,JC**,MR**,VG,RH**,DH**,AL		
	16-18	MP**,SV,KS**,JC,MR,VG**,RH	MP**,SV,KS,JC,MR,RH*,DH,AL*	SV,KS,JC,MR,RH,DH,AL	SV,KS,JC,MR,RH*,DH			
	18-20							
13 (ab 28.03.)	10-12	MP,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	ENDE	ENDE	ENDE
	12-14	MP,SV*,KS**,JC**,MR*,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	ENDE	ENDE	ENDE
	14-16	MP,SV,KS**,MR,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	ENDE	ENDE	ENDE
	16-18	MP,SV,KS**,MR,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	MP,SV,KS,JC,MR,DH,AL	ENDE	ENDE	ENDE
	18-20					ENDE	ENDE	ENDE

*) Nur eine Stunde
**) eingeplant aber nur bei Bedarf anwesend
***) Nur eine Stunde eingeplant, aber nur bei Bedarf anwesend (1+2)
?) Noch mit ... Fragezeichen

Edit This Page | Attach File | Delete This Version | Delete Page

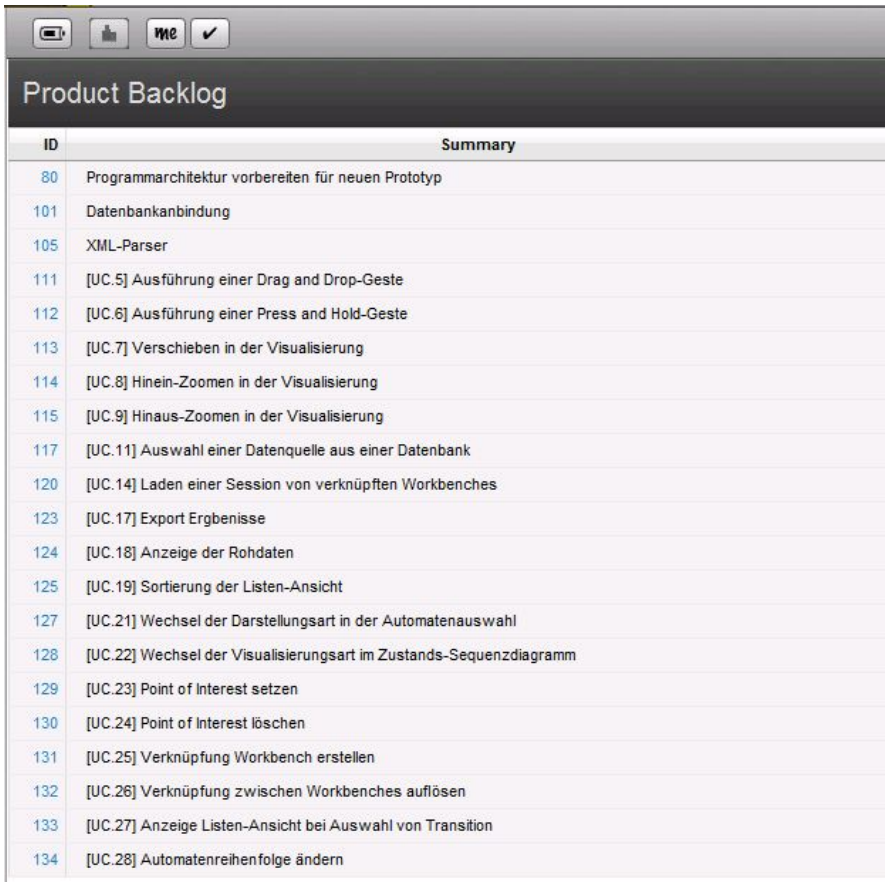
Abbildung 9.1: Zeitplanung im Wiki

Product Backlog

Die Funktion des Product Backlogs haben wir nach der Anforderungsanalyse genutzt um alle Anforderungen zu dokumentieren und geordnet zu speichern. Diese Liste wurde, ganz nach dem Gedanken der agilen Softwareentwicklung, ständig weiterentwickelt und verändert. Somit konnte eine fortlaufende Integration und Aufnahme aller Anforderungen sichergestellt werden. Abbildung 9.2 zeigt einen Ausschnitt des Product Backlogs zu einem Zeitpunkt des Projektes.

Scrum Backlog

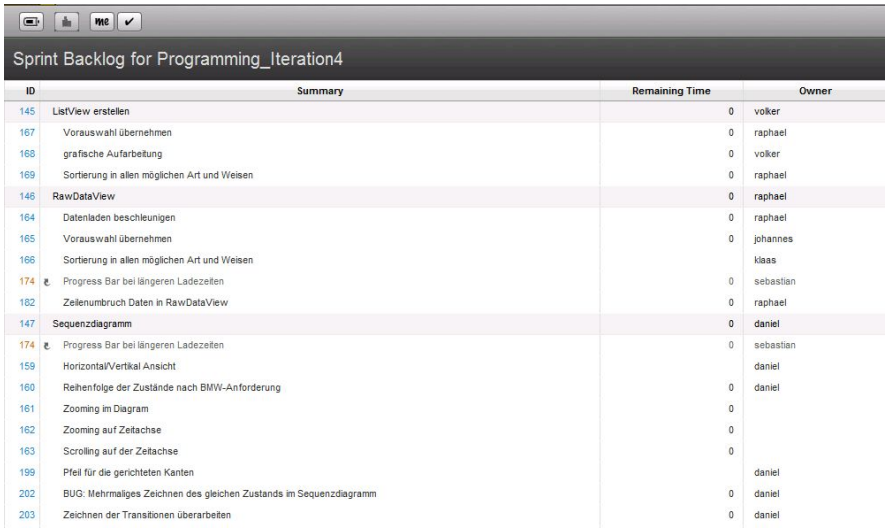
Das Scrum Backlog wurde für die Aufgabenverteilung der Iterationen genutzt. Die zu bearbeitenen Anwendungsfälle wurden aus dem Product Backlog für die jeweiligen Iterationen ausgewählt und mit Arbeitspaketen verfeinert. Somit konnte eine Zuordnung zu den jeweiligen pair pogramming Paaren stattfinden. Abbildung 9.3 zeigt eine Iteration aus dem Projekt. Da die Iteration jedoch schon abgelaufen ist, sind die Zeiten bei „Remaining Time“ überall auf null gesetzt.



Product Backlog

ID	Summary
80	Programmarchitektur vorbereiten für neuen Prototyp
101	Datenbankanbindung
105	XML-Parser
111	[UC.5] Ausführung einer Drag and Drop-Geste
112	[UC.6] Ausführung einer Press and Hold-Geste
113	[UC.7] Verschieben in der Visualisierung
114	[UC.8] Hinein-Zoomen in der Visualisierung
115	[UC.9] Hinaus-Zoomen in der Visualisierung
117	[UC.11] Auswahl einer Datenquelle aus einer Datenbank
120	[UC.14] Laden einer Session von verknüpften Workbenches
123	[UC.17] Export Ergbenisse
124	[UC.18] Anzeige der Rohdaten
125	[UC.19] Sortierung der Listen-Ansicht
127	[UC.21] Wechsel der Darstellungsart in der Automatenauswahl
128	[UC.22] Wechsel der Visualisierungsart im Zustands-Sequenzdiagramm
129	[UC.23] Point of Interest setzen
130	[UC.24] Point of Interest löschen
131	[UC.25] Verknüpfung Workbench erstellen
132	[UC.26] Verknüpfung zwischen Workbenches auflösen
133	[UC.27] Anzeige Listen-Ansicht bei Auswahl von Transition
134	[UC.28] Automatenreihenfolge ändern

Abbildung 9.2: Product Backlog



Sprint Backlog for Programming_Iteration4

ID	Summary	Remaining Time	Owner
145	ListView erstellen	0	volker
167	Vorauswahl übernehmen	0	raphael
168	grafische Aufarbeitung	0	volker
169	Sortierung in allen möglichen Art und Weisen	0	raphael
146	RawDataView	0	raphael
164	Datenladen beschleunigen	0	raphael
165	Vorauswahl übernehmen	0	johannes
166	Sortierung in allen möglichen Art und Weisen		klaas
174	Progress Bar bei längeren Ladezeiten	0	sebastian
182	Zeilenumbruch Daten in RawDataView	0	raphael
147	Sequenzdiagramm	0	daniel
174	Progress Bar bei längeren Ladezeiten	0	sebastian
159	Horizontal/Vertikal Ansicht		daniel
160	Reihenfolge der Zustände nach BMW-Anforderung	0	daniel
161	Zooming im Diagram	0	
162	Zooming auf Zeitachse	0	
163	Scrolling auf der Zeitachse	0	
199	Pfeil für die gerichteten Kanten		daniel
202	BUG: Mehrmaliges Zeichnen des gleichen Zustands im Sequenzdiagramm	0	daniel
203	Zeichnen der Transitionen überarbeiten	0	daniel

Abbildung 9.3: Scrum Backlog

Bug-Tracking

Das Bug-Tracking erfolgt über das integrierte Ticketing System. Somit kann jedes Projekgruppenmitglied Bugs direkt reporten und dem jeweiligen Verantwortlichen zuweisen. Abbildung 9.4 zeigt einen beispielhaften Bug-Report.

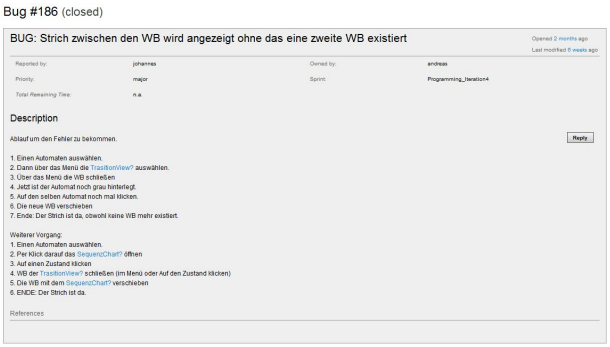


Abbildung 9.4: Bug-Reporting im Trac

Scrum Dashboard

Das Scrum Dashboard ist ein Mittel, welches uns ein direkte Übersicht über den Ablauf der Iteration darstellt. Das Burn Down Chart zeigt den Ablauf der Stunden in der Iteration. Durch die Anzeige des Trends kann ungefähr abgeschätzt werden, ob die Iteration zeitlich noch durchzuführen ist. Da das Dashboard sich jeweils direkt berechnet hat, konnten Analysen ohne extra Aufwand durchgeführt werden. Die folgende Abbildung zeigt einen Ablauf einer Iteration im Scrum Dashboard. Es ist auf dem Burn Down Chart gut zu erkennen, wie die Stundenzahl sich verringert.

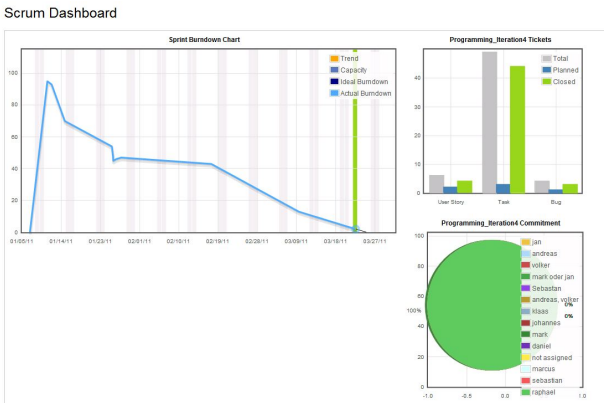


Abbildung 9.5: Scrum Dashboard

9.1.2 TortoiseSVN

Zur Benutzung von *Apache Subversion* (SVN) wird ein entsprechender Zugang benötigt. Eine Möglichkeit, diesen Zugang herzustellen stellt das Programm *TortoiseSVN* von Tigris.org dar. Es handelt sich dabei um einen Zugang mit grafischer Benutzerschnittstelle für Windows-Systeme. Nach der Installation kann der Benutzer direkt über das Kontextmenü des *Windows Explorer* beliebige Ordner als Ort für ein *SVN Repository* festlegen. Dazu muss der Benutzer lediglich die Adresse des zentralen Repository angeben, sich mit seinen Benutzerdaten authentifizieren und kann zukünftig das Repository nutzen. Dieser Ordner wird dann durch ein kleines Symbol erweitert, der die Aktualität der lokalen Kopie relativ zur Quelle des Repository angibt. Die Bedienung erfolgt vollständig über das Kontextmenü des Explorers, so dass das Starten weiterer Programme unnötig ist (s. Abb. 9.6). TortoiseSVN stellt zudem Hilfsmittel zur Verfügung, um beispielsweise ältere Versionen zu betrachten, oder Versionskonflikte zu lösen. Die Benutzung von TortoiseSVN steht unter der *GNU General Public License* und ist kostenfrei möglich [Tig11].

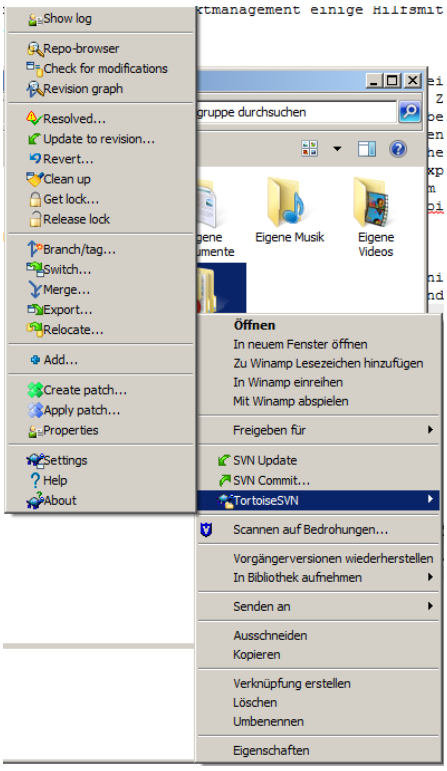


Abbildung 9.6: Bedienung von TortoiseSVN über das Kontextmenü des Windows Explorer

9.1.3 Doodle

Doodle ist ein Online-Werkzeug der Doodle AG zur Abstimmung von Terminen. Dabei kann jemand für ein bestimmtes Ereignis eine Reihe von Terminen zur Wahl stellen und diese Wahl dann per E-Mail an die gewünschten Teilnehmer versenden. Die Teilnehmer können dann die von ihnen präferierten Termine markieren und Kommentare verfassen. Durch diese Art der Abstimmung ist es leichter und schneller möglich, sich mit mehreren Leuten auf einen Termin zu einigen, als dies mit klassischen Methoden, wie zum Beispiel E-Mail, möglich wäre. Weder der Ersteller, noch die Teilnehmer einer Terminfindung müssen sich für diesen Vorgang registrieren und die Benutzung ist kostenlos [AG11a]. Wenn der Ersteller einer Terminfindung sich nicht registriert hat, muss er jedoch darauf achten, dass er den Verwaltungslink zu der Umfrage nicht verliert, da sonst die Umfrage nicht mehr bearbeitet werden kann. Außerdem hat ein registrierter Ersteller mehr Optionen bei der Gestaltung der Umfrage zur Auswahl.

9.1.4 LaTeX

LaTeX ist im eigentliche Sinne ein Softwarepaket zur vereinfachten Nutzung des Textsatzprogramms *TeX*. Für LaTeX gibt es eine Reihe von Bibliotheken, Übersetzern und Editoren. Die unter Windows wohl geläufigste Bibliothek ist *MikTeX*. Als Editoren für Windows bieten sich *TeXnicCenter* oder *Texmaker* an. Wobei ersteres den Nachteil hat, dass es nur die ISO-8859-15 Kodierung (auch *latin1* genannt) anbietet. Diese Tatsache sorgt oftmals für Probleme, wenn Benutzer von Linux oder MacOS mit Windows-Benutzern zusammen arbeiten wollen, da erstere in der Regel UTF-8 verwenden. Als reiner Editor ohne Übersetzer kann auch *Notepad++* sehr gut verwendet werden, da hier ebenfalls Syntaxhervorhebung für LaTeX (und viele weitere Sprachen) vorhanden ist.
LaTeX selbst wird oftmals als komplizierte Textverarbeitung missverstanden und benötigt eine recht hohe Einarbeitungszeit. Diese Tatsachen machen es oftmals schwer LaTeX als echte Alternative zu den populären Produkten wie *Microsoft Word* oder dem *OpenWriter* zu sehen. Allerdings zeigt sich die Schwäche der üblichen Office-Produkte im Zusammenhang mit der Verwendung von Versionierung, wie beispielsweise SVN, insbesondere dann, wenn mehrere Leute gleichzeitig an einem Dokument arbeiten. Bei LaTeX ist es jedoch problemlos möglich, ein Dokument aus mehreren Dateien zu erstellen. So kann, wie in diesem Bericht geschehen, jeder Abschnitt in eine eigene Datei geschrieben werden. Zur Verbindung aller Dateien muss eine Hauptdatei erstellt werden, in der festgelegt wird, wie und welcher Reihenfolge die einzelnen Teile zusammengehören. So kann jedes Gruppenmitglied seine Teile bearbeiten, ohne dass es zu Konflikten im SVN kommt.

9.2 Rollenverteilung

Zu Beginn der Projektgruppe wurden bestimmte Rollen festgelegt, von denen jedes Gruppenmitglied eine erfüllen sollte. Eine Auflistung (vgl. Abbildung 9.7) und kurze Beschreibung der Rollen erfolgt in diesem Abschnitt.

Bezeichnung Rolle	Anzahl Plätze	Ausführende Personen
Projektmanager	2	Sebastian Vehring, Klaas Schmidt
Dokumentationsbeauftragter	1	Andreas Löcken
Chief Designer	1	Marcus Rehnen
Chief Programer	1	Daniel Häuser
Usability Engineer	1	Johannes Cordes
Quality Engineer	1	Raphael Holtmann
Forscher	2	Volker Gollücke, Mark Phlippen
Administrator	1	Jan Kirchhoff

Abbildung 9.7: Tabelle Rollenstruktur Projektgruppe

9.2.1 Projektmanager

Die Aufgabenbereiche des Projektmanager erstrecken sich größtenteils in den Bereichen der Organisation, Führung und Kontrolle der Gruppe. Die Aufgaben bestehen hierbei im Einzelnen aus:

- Durchführung und Einhaltung des Vorgehensmodells für die Projektgruppe
- Moderation der Gruppenmeetings
- Planung von besonderen Events oder anderen Tätigkeiten
- Iterationsplanung und -vorbereitung
- Überwachung des Meilensteinplans
- Schnittstelle zu Gruppenbetreuern
- Planung und Koordination von allgemeinen Aufgaben, die nicht einer anderen Expertenrolle zugeordnet werden können.

9.2.2 Dokumentationsbeauftragter

Das Aufgabenfeld des Dokumentationsbeauftragten erstreckt sich über sämtliche Bereiche, in denen irgendeine Form von Dokumentation verfasst werden muss. Hierbei steht insbesondere die Organisation und Koordination im Vordergrund. Im Einzelnen heißt das:

- Ausarbeiten von Vorlagen, sofern benötigt
- Fortschritt und Form der Enddokumentation und Zwischenergebnisse in Zusammenarbeit mit dem Quality Engineer überprüfen. Dazu zählt insbesondere:
  - Einheitliches Aussehen der Dokumente
  - Einheitlicher Befehlssatz in LaTeX, beispielsweise beim Einbinden von Grafiken
  - Lösen von Problemen, die sich zum Beispiel aus dem Umgang mit LaTeX ergeben

9.2.3 Chief Designer

Dem Chief Designer fällt es zu, sich um die optische Präsentation der Anwendung zu kümmern. Dies beinhaltet im Einzelnen:

- Erstellung und Umsetzung von
  - Design Richtlinien und Grafiken
  - Farbwahl
  - Rundungen
  - Ästhetik und Bedienung in Zusammenarbeit mit dem Usability Engineer
  - Zuerst Design festlegen, dann Bedienung anpassen
- Gestalten
  - Entwicklung des Interaktionskonzepts
  - Visuelle Gestaltung der Benutzeroberfläche
- Visualisierung und Prototyping
  - Grafische Aufbereitung der Gestaltungslösung oder Umsetzung als einfacher Prototyp
- Empirische Evaluation

  - Überprüfung der Gestaltungsideen hinsichtlich der Erfüllung der definierten Aufgaben
  - Verbesserungsmöglichkeiten daraus ableiten

9.2.4 Chief Programer

Die Arbeit des Chief Programer umfasst vor allen Dingen sämtliche Aspekte, die mit der Programmierung zu tun haben. Dazu zählt insbesondere:

- Einhaltung des Vorgehensmodells überprüfen und nötigenfalls lenkend eingreifen
- Zusammen mit dem Qualitiy Engineer Programmierrichtlinien erstellen
- Ansprechpartner für allgemeine, die Programmierung betreffende Probleme

9.2.5 Usability Engineer

Der Usability Engineer hat vor Allem die Aufgabe, die Anwendung auf Benutzbarkeit hin zu überprüfen und für gute Benutzbarkeit zu sorgen. Dazu zählt insbesondere:

- Durchführung von Nutzbarkeitsevaluationen
- Beachtung der Softwareergonomie
- Designs mit Nutzbarkeitsrichtlinien abgleichen
- Mitgestaltung der Designrichtlinien

9.2.6 Quality Engineer

Im Aufgabenbereich des Quality Engineer liegt die Qualität der Anwendung und aller weiteren Erzeugnisse der Projektgruppe. Es handelt sich dabei um eine Querschnittsfunktion, die Einblick in alle Bereiche hat. Dazu zählt unter anderem:

- Vorgabe von Qualitätsrichtlinien für Programmierrichtlinien
- Kontrolle über das Durchführen von Tests
- Qualitätskontrolle der Code-Dokumentation
- Qualitätskontrolle des Abschlussberichtes und sonstiger Dokumente

9.2.7 Forscher

Die Forscher haben vor allem zwei Aufgabenbereiche. Dazu zählt zum Einen, die Forschungsergebnisse der Gruppe für die Publikation vorzubereiten und zum Anderen, die Entwicklungen im Bereich des Arbeitsschwerpunktes der Gruppe im Auge zu behalten. Das heißt im Detail:

- Publikationen der Gruppe koordinieren
  - Rahmenbedingungen dafür schaffen
  - Auf Einhaltung dieser achten
  - Verteilung von Teilaufgaben an Gruppenmitglieder
- Entwicklungen im Auge behalten
  - Welche Entwicklungen gibt es im Bereich MultiTouch, Surface Computing und visuelle Analyse?
  - Wie kann die Gruppe davon profitieren?

9.2.8 Administrator

Die Einrichtung, Pflege und Wartung der Serverdienste für die Gruppe sind die Aufgaben des Administrators. Das heißt im Einzelnen:

- Beschaffen der Benutzerdaten der Gruppenteilnehmer
- Beschaffung der Serverdienstleistungen von der ARBI
- Einrichten und Verwalten von:
  - SVN
  - Apache
  - MySQL
  - Python local mirror
  - Trac und Plugins
- Bereitstellen des SVN
- Skripte zum (Neu)Starten der Serverdienste bereithalten
- Überwachung der Serverprozesse und -logs
- Wartung des Tisches

9.3 Projektablauf

Die Zeitplanung des gesamten Projektes wurde in zwei Releases gegliedert, welche sich anhand der Semesterstruktur gegliedert haben. Zudem gab es Meilensteine die wichtige Projektereignisse definiert haben. Die folgende Auflistung 9.8 zeigt die Projektstruktur und die jeweiligen Start und Endzeitpunkte der Vorgänge, welche anschließend auch nochmal genauer erläutert werden. Der genaue Projektablauf ist auch nochmal genauer in dem folgenden Gantt-Chart(s. Abbildung 9.9) ersichtlich.

Vorgangsname	Dauer	Anfang	Ende	Vorgänger
Start Projekt	0 Tage	Do 01.04.10	Do 01.04.10	
Start 1. Release	**140 Tage**	**Fr 02.04.10**	**Do 14.10.10**	**1**
Seminararbeitsphase	**25 Tage**	**Fr 02.04.10**	**Do 06.05.10**	**1**
TOAD-Projekt	**107 Tage**	**Mi 19.05.10**	**Do 14.10.10**	**3**
Anforderungsermittlung	**84 Tage**	**Mi 19.05.10**	**Mo 13.09.10**	**5**
Konzept nach Sciva erarbeiten	**23 Tage**	**Di 14.09.10**	**Do 14.10.10**	**7**
Vast-Challange	**85 Tage**	**Fr 11.06.10**	**Do 07.10.10**	**3**
Ende 1. Release	0 Tage	Do 14.10.10	Do 14.10.10	27;6;2;18
Start 2. Release	**120 Tage**	**Fr 15.10.10**	**Do 31.03.11**	**33**
Toad-Projekt	**110 Tage**	**Fr 15.10.10**	**Do 17.03.11**	**33**
Umsetzungsphase	**110 Tage**	**Fr 15.10.10**	**Do 17.03.11**	**33**
Abschlussdokumentation	**64 Tage**	**Mo 03.01.11**	**Do 31.03.11**	**33**
Abschluss Gesamtprojekt	0 Tage	Do 31.03.11	Do 31.03.11	34;35;45;36

Abbildung 9.8: Projektablauf tabellarisch

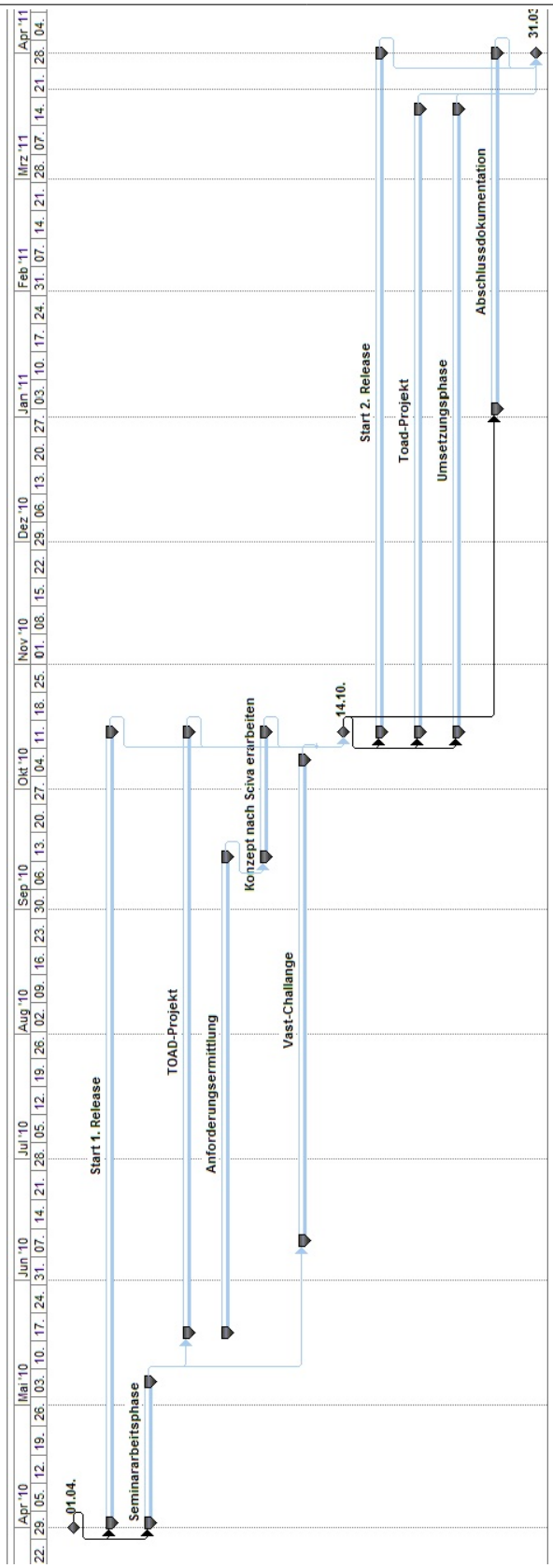


Abbildung 9.9: Projektablauf Gantt-Chart

9.3.1 Release 1

Das erste Release stellte in unserer Projektgruppe das erste Semester dar und beinhaltete die folgenden Arbeiten. Diese werden in den Unterkapiteln genauer erläutert. Als Arbeitsergebnisse werden das Konzept über die Anforderungen von BMW bezüglich des TOAD-Prototypen sowie der Prototyp für die VAST-Challenge angesehen.

Seminararbeitsphase

In der Projektarbeitsphase musste jedes Mitglied eine Seminararbeit anfertigen. Diese Arbeiten stellten die grundlegende Einarbeitung in den gesamten Themenkomplex dar. Dadurch, dass diese Arbeiten am Anfang des Projektes erfolgten, konnte eine schneller Einstieg ins Thema gefunden werden. Folgend werden die Themen der Seminararbeiten dargestellt.

- Multitouch-Technologien
- Visual Analytics auf Surface Computern
- Agile Software-Entwicklung
- Agiles Projektmanagement
- BMW Datalogs
- Usability Engineering
- Visual Analytics: Human Factors, Requirements
- Visual Analytics: State-Of-The-Art
- Visual Analytics : Softwaretechnisch, Vis-Frameworks
- Visual Analytics: Visualisierungsdesign

TOAD-Projekt Anforderungsermittlung / Konzepterarbeitung

In diesem ersten Part des TOAD-Projektes wurden die Anforderungen mit BMW ermittelt und protokolliert. Für den direkten Austausch dienten Telefonkonferenzen, auf denen unter anderem über Screen-Sharing die Ergebnisse präsentiert wurden. Nach einer ersten allgemeinen Analyse der Anforderungen wurde zielgerichtet mit dem SCIVA-Prozess weiter analysiert und ein Konzept ausgearbeitet, welches als Grundlage für die spätere Programmierung diente. Die gesamte Phase der Anforderungsermittlung und Konzeptausarbeitung zeigt folgende Abbildung 9.10.

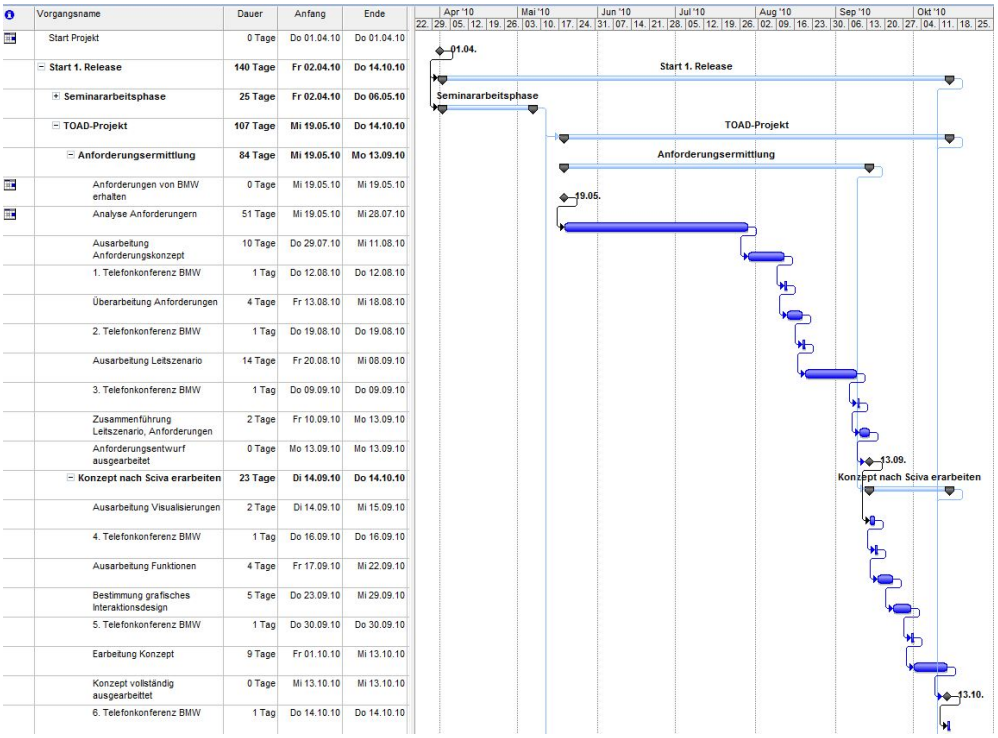


Abbildung 9.10: Ablauf Anforderungsermittlung TOAD

VAST-Challenge

Die VAST-Challenge (Challenge des VAST-Symposium 2010) diente uns als Datengrundlage für einen ersten Prototypen, welcher für die Einarbeitung in die Programmierung genutzt wurde. Es wurden dabei epidemiologische Daten untersucht und ausgewertet. Für den Prototyp wurde eine Anforderungsanalyse sowie eine Priorisierung der Anforderungen durchgeführt. Anschließend erfolgte die Umsetzung in mehreren Iterationen. Dieses Vorgehen ermöglichte erste Erfahrungen im Bereich der agilen Entwicklung, welche später für die eigentliche Umsetzung des TOAD-Projekts genutzt werden konnten. Es wurde zudem erste Erfahrungen mit dem WPF-Framework sowie mit der Programmierung von Multitouch-Applikationen gesammelt.

Folgende Abbildung 9.11 zeigt das Ergebnis des Prototypen, welcher teilweise auch als Grundlage für die spätere Programmierung im TOAD-Projekt genutzt wurde.

In der Abbildung 9.11 werden zwei Workbenches gezeigt. Auf einer ist die Filterung der Eingangsdaten zu sehen und auf der anderen ist eine mögliche Analyse mittels eines Pie-Charts abgebildet.

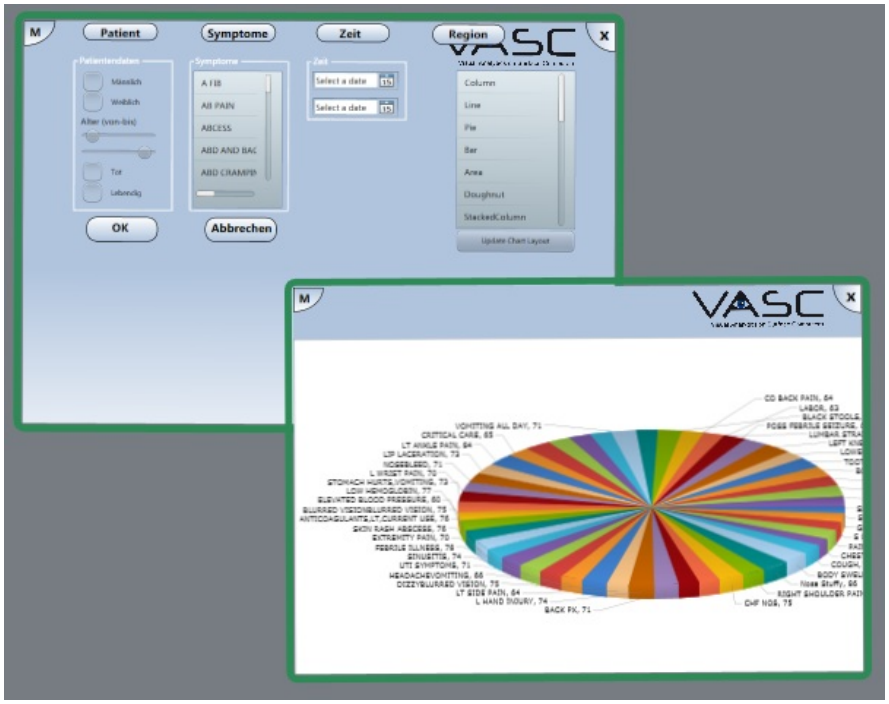


Abbildung 9.11: VASC-Prototyp

9.3.2 Release 2

Im zweiten Release wurde das Konzept für das TOAD-Projekt umgesetzt. Dies erfolgte durch die Programmierung eines Prototypen, welcher die gesetzten Anforderungen abdeckt. Zudem wurde die Abschlussdokumentation angefertigt.

TOAD-Projekt Umsetzungsphase

In der Umsetzungsphase wurden die Anforderungen gemäß ihrer Priorisierung in Iterationen eingeteilt und anschließend umgesetzt. Dies erfolgte nach dem bereits vorgestellten adaptierten Vorgehensmodell. Somit konnte ein rascher Fortschritt erzielt werden. Zudem konnten noch einige Komponenten aus dem VASC-Challenge-Prototypen übernommen werden. Nebenläufig zur Umsetzung wurden zwei Evaluationen durchgeführt. Die Ergebnisse hielten, aufgrund des agilen Vorgehens, direkt im Entwicklungsprozess Einzug. Somit konnte eine stetige Verbesserung erreicht werden. Bei der abschließenden Präsentation bei BMW in München hat das Programm seine „Feuertaufe“ bestanden.

Kapitel 10

Entwicklungsparadigmen

Zu Beginn der Softwareprogrammierung hat sich die Projektgruppe auf einheitliche Coding-Konventionen geeinigt. Die Konventionen gliedern sich in die Kategorien Kommentierung, Deklarationen, Statements, Zeichen- und Zeilenabstand, Namenskonventionen sowie Programmiertechniken. In der Kategorie Namenskonventionen wurde beispielsweise beschrieben, dass Variablen nach Camel Casing (eine Namenskonvention nach der keine Trennzeichen verwendet werden, das erste Wort mit einem Kleinbuchstaben beginnt und alle weiteren mit einem Großbuchstaben beginnen) und Methoden nach Pascal Casing (eine Namenskonvention nach der keine Trennzeichen verwendet werden und alle Wörter mit einem Großbuchstaben beginnen) benannt werden. Weiterhin wurde hier gesagt, dass alle Klassenvariablen mit einem vorangestelltem Unterstrich benannt werden und alle Konstrukte einen englische Namen erhalten.

Zur Umsetzung der definierten Coding-Konventionen wurde die Visual Studio Erweiterung ReSharper in der Version 5 benutzt (http://www.jetbrains.com/resharper). ReSharper ergänzt die Funktionalität der Visual-Studio-Entwicklungsumgebung vor allem in den Bereich Code-Refactoring und Code-Navigation. Das Ziel von ReSharper ist eine Produktivitätssteigerung beim Entwickeln. Da unsere Projektgruppe ein studentisches Projekt ist, konnten wir kostenlos eine einjährige Lizenz vom Hersteller benutzen.

Die erstellten Coding-Konventionen wurden in eine sogenannte ReSharper-Settings-Datei im XML-Format überführt und konnten anschließend von jedem Projektgruppenteilnehmer in die Entwicklungsumgebung importiert werden. Durch den Import der Einstellungen wird das Refactoring durch ReSharper bei allen Projektgruppenteilnehmern nach den gleichen Konventionen durchgeführt. Weiterhin wird der Entwickler darauf aufmerksam gemacht, wenn sein Code nicht den definierten Konventionen entspricht.

Teil III

TOAD - Kollaborative visuelle Analyse von Busdaten aus Fahrzeugen

Kapitel 11

Projektbeschreibung

In diesem Kapitel wird beschrieben, welche Ziele die Anwendung verfolgen soll. Des Weiteren wird ein Beispiel-Szenario vorgestellt, von der Datengewinnung bis zur Auswertung dieser Daten. Im darauf folgenden Unterkapitel wird die Systemumgebung beschrieben, auf dessen Grundlage die Anwendung entwickelt wurde. Zu guter Letzt werden die Iterationen mit ihren zugehörigen Arbeitspaketen dargestellt und die Kommunikationswege zwischen den Projektbeteiligten vorgestellt.

11.1 Zielbeschreibung

In diesem Kapitel werden die Anforderungen beschrieben, welche BMW an die neue Software stellt. Dadurch ergeben sich die an uns gestellten Ziele. Um schneller, die bei den Testfahrten aufgetretenen, Fehler zu finden, benötigt BMW Darstellungsformen bzw. eine Software um die jeweiligen Testfiles zu analysieren. Diese Anwendung sollte zudem die kollaborative Arbeit unterstützen und die Datenbereiche auf verschiedenen Abstraktionsebenen darstellen. Dies sind zudem die Hauptziele der neuen Software. Um dieses Ziel zu lösen, entstand die Idee der Umsetzung auf einem Tisch mit Multitouch-Bedienung, welcher sich mittels Gesten ansteuern lässt. An diesem Tisch sollen verschiedene Anwender gleichzeitig arbeiten können, um somit schneller ihr Wissen und ihre Erfahrungen mit anderen austauschen zu können. Dazu sollen, die mit Hilfe der Software *Carmen* generierten, Daten (XML Files mit Rohdaten und Transitionsdaten der Automaten) in das System eingelesen werden und visuell aufbereitet dargestellt werden. Carmen ist eine Sammlung von Werkzeugen, um Messungen am Bus-System eines Fahrzeuges durchzuführen.

11.2 Szenariobestimmung

Das folgende Szenario soll einen Arbeitsablauf der Testfahrt, mit Aufnahme der Bus-Daten, bis zur Analyse dieser darstellen. Dies Ziel liegt in der Findung eventueller Fehler.

Annahme für das Szenario:

Bei einer Testfahrt wird der Fehler festgestellt, dass ein Fenster sich nicht wieder schließen lässt, nachdem es geöffnet wurde. Dieser Fehler soll anhand der Bus-Daten analysiert und behoben werden.

11.2.1 Schritt 1: Testfahrt und Aufzeichnung der Kommunikation

Während einer Testfahrt werden die Daten, welche über die verschiedenen Bus-Systeme gesendet werden, aufgezeichnet. Diese Aufzeichnung wird mit dem Tool Carmen im Online-Modus durchgeführt. Die aufgezeichneten (Roh-) Daten werden in sogenannten Trace-Files gespeichert, welche in zeitlicher Abfolge die über das Bus-System versendeten Nachrichten darstellen. Die für die Analyse relevanten Bus-Daten können auch direkt mit Hilfe von Automaten gefiltert und somit vorverarbeitet werden. Neben der direkten Aufzeichnung (Online-Modus) wird auch das nachträgliche abspielen aufgezeichneter Testfahrten im Offline-Modus ermöglicht (Simulation). Dazu werden die Trace-Files, welche im Online-Modus gespeichert wurden, erneut eingelesen und können ebenfalls mit Hilfe von Automaten gefiltert werden.
Automaten werden zur Analyse der Daten genutzt und können in Abhängigkeit zu den Rohdaten ihren Zustand ändern. Der Zustand des Automaten gibt Auskunft über den Zustand der (Roh-)Daten. Dabei kann der Zustand folgende Typen aufweisen: WARN, DEFECT, INFO oder OK. Die Automaten müssen im Bezug auf eine konkrete Fragestellung manuell erstellt werden. Die Daten können in verschiedenen Formaten gespeichert werden. Einerseits ist es möglich die Rohdaten separat zu speichern und andererseits können die Rohdaten mit den Transitionsdaten aus den Automaten kombiniert werden. In unserem Szenario wurde bei der Testfahrt festgestellt, dass sich das Fenster nicht schließen lässt. Hierzu werden die Rohdaten genauer analysiert. Folgende Automaten werden gebildet:

- Automat zur Überwachung der Motordrehzahl
- Automat zur Überwachung des Fensterhebers auf der Fahrerseite
- Automat zur Überwachung des Zustandes der Klimaanlage

Diese Automaten werden im Offline-Modus mit den Rohdaten verknüpft und die Ergebnisse der Automaten werden inklusive der Rohdaten in eine XML-Datei exportiert.

11.2.2 Schritt 2: Analyse der Automatenausgaben

Nach dem Erstellen der XML-Dateien durch Carmen müssen diese genauer analysiert werden. Hierzu gibt es verschiedene Möglichkeiten, welche abhängig vom Ziel der Analyse und den Vorlieben der Analysten sind. Nachfolgend werden alle Möglichkeiten im Bezug auf das Beispielszenario vorgestellt:

Cardiogramm

Cardiogramm ist ein Flash-Tool zur Darstellung von Automaten in zeitlicher Abfolge. Zudem können mehrere Automaten untereinander angeordnet und abgeglichen werden. Für das Szenario wird die XML-Datei in das Programm geladen und es erfolgt die Anzeige aller Automaten für den gleichen Zeitabschnitt. Durch die Anzeige aller drei Automaten lässt sich folgendes Muster erkennen:
Zum Zeitpunkt des Fehlerzustands im Automaten zur Überwachung des Fensterhebers auf der Fahrerseite wechselte der Automat für die Motordrehzahl in den Warnungszustand, da die Drehzahl über 3000UpM stieg. Außerdem befand sich der Automat für die Klimaanlage im Zustand OK, da die Klimaanlage eingeschaltet war und fehlerfrei funktionierte. Daraus kann der Analyst vermuten, dass es einen Fehler im Zusammenhang zwischen der Motordrehzahl und der Fensterheber gibt. Es muss nun eine weitere Analyse dieser beiden Automaten erfolgen.

Excel (XLS-Dateien)

Bei der Analyse mit der Microsoft Tabellenkalkulation Excel werden die XML-Dateien in eine Tabellenstruktur überführt. Zur Unterstützung der präattentiven Verarbeitung wurde die Tabellenstruktur durch eine Farbkodierungen ergänzt. Durch geschicktes Filtern und Scrollen in der Tabelle kann der Analyst den gleichen Sachverhalt wie im Kardiogramm herausfinden.

Rohdatenanalyse (CBL-Dateien)

In den Rohdaten ist der Fehler für den Analysten nicht ersichtlich. Nachdem der Fehler mit den Tools Excel oder Cardiogramm gefunden wurde, kann jedoch eine erweiterte Analyse mit den Rohdaten erfolgen.

XML-Datei Analyse

Eine weitere Analysemöglichkeit ist der Vergleich von zwei XML-Dateien. Dies kann sinnvoll sein, wenn der Fehler durch das Aufspielen einer neuen Firmware das erste Mal auftritt und bei alten Testfahrten noch nicht aufgetreten ist. So kann ein direkter Vergleich zwischen den alten und den neuen Trace-Files erfolgen. Zusätzlich können XML-Files

direkt mit einem entsprechenden Anzeige-Tool analysiert werden. Im Gegensatz zu den Rohdaten können die XML-Files vom Analysten direkt interpretiert werden.

11.3 Systemumgebung

In diesem Abschnitt wird auf die im Projekt verwendete Hardware und Software eingegangen. In Abbildung 11.1 ist der vom OFFIS entwickelte Multitouch-Tisch abgebildet, auf dem das System entwickelt wird. In dem Tisch befindet sich ein Projektor (b), der von unten an die Oberseite des Tisches strahlt. Diese Projektionsfläche (c) wird seitlich von Infrarot-LEDs (d) angestrahlt. Durch Berührungen wird das Infrarotlicht so gebrochen, dass eine Kamera (e) mit Infrarotfilter unter der Projektionsfläche das Licht auffängt.

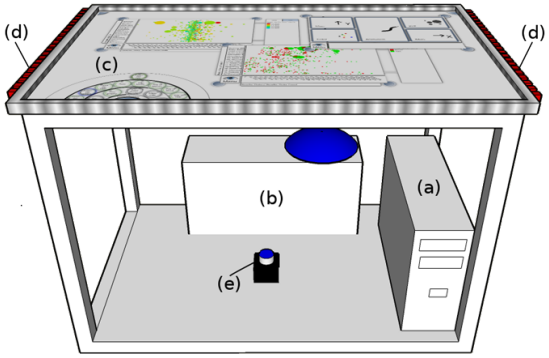


Abbildung 11.1: Multitouch-Tisch

Der eingebaute PC (a) verarbeitet die Kamerabilder mit Hilfe der Software *Community Core Vision* zu Nachrichten des TUIO-Protokolls, die von *Multi-Touch Vista* in Touch-Signale für Microsoft Windows 7 aufbereitet werden. Die wichtigsten Bauteile des Rechners sind ein Intel Core 2 Quad Q9400 4 x 2.67 GHz mit 8 GB Arbeitsspeicher und einer NVIDIA GeForce 9800 GT Grafikkarte.

Der Tisch hat eine Bilddiagonale von 145cm (ca. 57 Zoll) und eine Auflösung von 1280 x 800 Pixeln. Außerdem hat der Tisch die Maße 120 x 100 x 80cm (BxHxT).

Das System arbeitet mit dem Betriebssystem Microsoft Windows 7, jedoch können auch andere Versionen genutzt werden. Dies wird jedoch nicht getestet. Das Hauptargument für Windows 7 ist die native Multi-Touch Unterstützung. Als Programmiersprache kommt C# zum Einsatz, da dort das Erstellen von Benutzerschnittstellen mit Touch-Erkennung durch die WPF-Unterstützung(Kapitel 2.5.2) vereinfacht wird.

11.4 Iterationen

Das Projekt wurde insgesamt in 6 Iterationen (Definition in Kapitel 8.1 „Agiles Projektmanagement“) aufgeteilt, welche wiederum mehrere Arbeitspakte beinhalteten. Der Bearbeitungszeitraum einer Iteration erstreckte sich auf einen Zeitraum von vier bis sechs Wochen. Die Arbeitspakete wurden vor dem Beginn der Implementierung ermittelt und priorisiert. Somit konnte die Reihenfolge der Arbeitspakete ermittelt werden und den Iterationen zugeordnet werden. Nachdem eine Iteration abgeschlossen war, wurden die Arbeitspakete ein weiteres Mal priorisiert. Dies geschah, wenn nötig, auf Grundlage von neuen Erkenntnissen während der Implementierung der vorherigen Iteration. Die Priorisierungen wurden bei Meetings mit den Projektleitern, sowie dem ganzen Projektteam durchgeführt. Dabei wurden die einzelnen Anwendungsfälle, welche im Anhang A aufgeführt sind, durch die Anwesenden auf einer Skala von 1-5 bewertet. Durch die Berechnung des Mittelwerts wurde die Wichtigkeit des Arbeitspakets eingestuft. Die Anwendungsfälle wurden zu Anfang des Projekts zusammen mit BMW ausgearbeitet und im Laufe des Entwicklungszeitraums nicht eins zu eins übernommen. Zum Beispiel wurde in einigen Anwendungsfällen erwähnt, dass zum Ausführen einer bestimmten Aktion das TaP-Menü geöffnet werden sollte, da zu Anfang die Idee bestand, dieses fertig entwickelte Menü in unser Projekt einzubinden. Im Laufe der Entwicklung entschieden wir uns aber, ein eigenes Marking-Menü zu entwickeln.

Im Folgenden werden die Iterationen mit ihren zugehörigen Arbeitspaketen dargestellt:

Iteration 1 02.07.2011 - 03.08.2010

- MVVM Struktur einführen
- Initialisierung der Anwendung
- Erstellung einer Workbench

Iteration 2 03.08.2010 - 21.09.2010

- Workbenchhandling
- Daten importieren aus XML
- Automatenansicht

Iteration 3 21.09.2010 - 01.11.2010

- History-Funktion
- Aus- und abwählen von Automaten
- Zoomen in Automatenansicht

Iteration 4 01.11.2010 - 20.12.2010

- Zustandssequenzdiagramm
- Workbenches verknüpfen
- Zoomen in Zustandssequenzdiagramm

Iteration 5 20.12.2010 - 01.02.2011

- Transitionliste
- Rohdatenliste
- Session speichern/ laden
- Marking-Menü

Iteration 6 01.02.2011 - 12.03.2011

- Filter-Funktion
- Workbench Synchronisation
- Ergebnis Export

Dies war eine grobe Gliederung der Arbeitspakete, welche wiederum in kleinere Arbeitspakete untergliedert wurden. Beispielsweise wurde das Paket *Workbenchhandling* in die folgenden Unterpakete eingeteilt:

- Workbench verkleinern
- Workbench vergrößern
- Workbench rotieren
- Workbench verschieben
- Workbench öffnen
- Workbench schließen

Während der einzelnen Iterationen sind außer den Arbeitspaketen weitere Aufgaben angefallen, wie die Aufgabenanalyse und -spezifikation, weitere Implementierungen, sowie das Testen des geschriebenen Codes. Die Visualisierungen und Funktionen wurden während des gesamten Entwicklungsprozesses optimiert. Durch die zyklische Rücksprache mit den Projektleitern und BMW ergaben sich permanent Optimierungsvorschläge. Im folgenden Kapitel werden die Kommunikationswege mit BMW beschrieben, sowie ein Beispiel dargestellt, wie die ständige Kommunikation unsere Arbeit positiv beeinflusste.

11.5 Kommunikation

Die stetige Kommunikation mit dem Projektpartner BMW war wichtig für die Optimierung der Anwendung. Es sind verschiedene Kommunikationswege zum Einsatz gekommen. Zu Anfang wurden *Telefonkonferenzen* mit den Verantwortlichen von BMW organisiert und durchgeführt, um den Anforderungskatalog zu spezifizieren. Die Projektgruppe traf sich dazu in einem Konferenzraum mit einem Telefon welches eine Freisprecheinrichtung besaß. Die Konferenz wurde durch eine Person des Projektmanagements geleitet und durchgeführt. Im weiteren Projektverlauf wurde die Anwendung WebEx genutzt, um *Webkonferenzen* abzuhalten. Folgend sind Bilder einer Webkonferenz dargestellt.

Abbildung 11.2: Webkonferenz

Dabei hatten wir zusätzlich die Möglichkeit, neben der sprachlichen Kommunikation, auch die visuelle Darstellung zu nutzen. Dazu wurden Präsentationen mit den aktuellen Entwicklungsständen und weiteren Informationen angefertigt, die während der Konferenz per Screensharing vorgestellt wurden. So konnten die Ingenieure von BMW direkt konstruktive Kritik zu aktuellen Entwicklungen geben. Im der Abbildung 11.3 wird anhand der Automatenansicht illustriert, welchen Einfluss die Kommunikation auf diese Visualisierung hatte.

Der erste Vorschlag der Automatenansicht wurde mittels eines Bildbearbeitungsprogramms erstellt (1) und am 14.10.2010 während der Webkonferenz vorgestellt. Durch Verbesserungsvorschläge der Ingenieure von BMW wurde schließlich die zweite Visualisierung (2), wiederum mit einem Bildbearbeitungsprogramm, erstellt

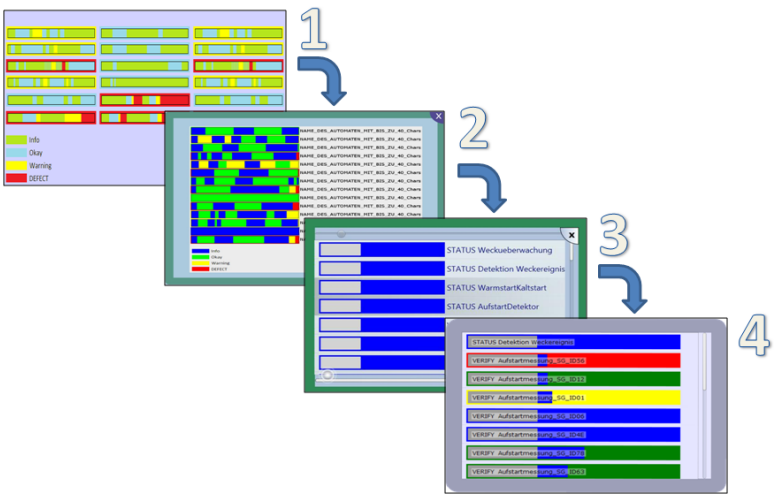


Abbildung 11.3: Entwicklung der Automatenansicht

und während der nächsten Konferenz vorgestellt. Aus den daraus resultierenden Vorschlägen wurde die Automatenansicht zum ersten Mal prototypisch implementiert (3). Durch weitere Verbesserungsvorschläge seitens der Ingenieure und innerhalb des Projektteams entsprang schließlich die endgültige Automatenansicht (4). Zum fortgeschrittenen Entwicklungsstand des Prototyps wurden außerdem Videos angefertigt, um den Ingenieuren von BMW den aktuellen Prototyp vorzuführen und so Feedback zum Interaktionsdesign zu erlangen. Diese Videos wurden ebenso per Screensharing über WebEx vorgestellt.
Durch regen E-Mail-Verkehr zwischen BMW und dem Projektteam wurden ebenso einige Fragen beantwortet und Probleme gelöst. Die folgende Abbildung11.4 zeigt die wichtigsten Korrespondenzen während der gesamten Projektphase.

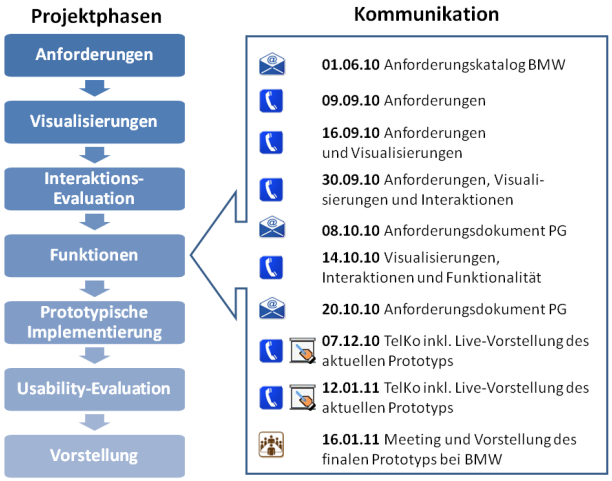


Abbildung 11.4: Informationsaustausch

Kapitel 12

Projektdurchführung

12.1 Anforderungen

Im Folgenden werden die funktionalen und nichtfunktionalen Anforderungen an das System vorgestellt.

12.1.1 Funktionale Anforderungen

Das System benötigt eine Vielzahl von bestimmten Funktionen. Die genauen Anforderungen an diese Funktionen sind im folgenden Abschnitt festgehalten.

Allgemeine funktionale Anforderungen

Gestische Bedienung Der Anwender soll die Möglichkeit haben, so direkt wie möglich mit dem Programm bzw. dessen Bestandteilen zu interagieren. Die Bedienung soll daher sowohl auf Single- als auch auf Multi-Touch-Gesten ausgelegt sein.

Vollbildmodus Um den maximal möglichen Platz für die Analyse ausnutzen zu können, soll die Benutzungsoberfläche des Programms stets im Vollbildmodus ausgeführt werden.

Anforderungen an die Datenhaltung

Unterstützung großer Datenmengen Aufgrund des immensen Datenvolumens der Trace-Files ist es notwendig, dass das System große Datenmengen unterstützt und visualisieren kann.

Vordefinierte XML-Strukturen Die einzuladenden Trace-Files werden mit Hilfe von Carmen erzeugt und besitzen eine feste XML-Struktur bestehend aus Automatentransitionen und Rohdaten. Die Anwendung muss in der Lage sein diese Daten laden und visualisieren zu können.

Anforderungen an die Workbenches

Zur gemeinsamen Arbeit an dem System werden Workbenches benötigt, die frei auf dem Multi-Touch-Tisch angeordnet werden können.

Workbenchhandling Im System müssen sich Workbenches erstellen, schließen, skalieren, rotieren und verschieben lassen, um den Analysten bei der Anordnung die größtmögliche Freiheit zu gewähren.

Multiple Coordinated Views Darüber hinaus wollen die Analysten mehrere Sichten auf die Daten parallel und nebenläufig haben, um die Daten aus verschiedenen Sichten analysieren zu können. Beispielsweise soll es möglich sein in einer View die Automatentransitionen zu analysieren und sich in einer anderen View die zugehörigen Rohdaten anzeigen zu lassen.

Anforderungen an die Datenauswahl und Datenspeicherung

Laden von Trace-Files Die Anwendung muss in der Lage sein beliebige Trace-Files zu laden. Beim Laden einer Trace-File werden die XML-Daten interpretiert und in die Anwendung geladen.

Speichern einer Sitzung Eine begonnene Sitzung soll gespeichert werden können. Beim Speichern der Sitzung werden die aktuellen Analyseschritte und eine Referenz auf die Trace-File gespeichert. Die Sitzung wird lokal auf dem Computer gespeichert und alle gespeicherten Informationen liegen hierbei im XML-Format vor.

Laden einer Sitzung Eine gespeicherte Sitzung soll wieder aufgerufen werden können. Durch das Laden einer Sitzung wird zunächst die referenzierte Trace-File geladen und anschließend werden die gespeicherten Analyseschritte durchgeführt. Vor dem Laden einer Sitzung muss sichergestellt werden, dass die referenzierte Trace-File am erwarteten Speicherort abgelegt ist.

Anforderungen an den Analyseprozess

Zusammenhänge darstellen Die Anwendung muss in der Lage sein zeitliche Zusammenhänge innerhalb von Automatentransitionen und den Rohdaten anzeigen zu können. Beispielsweise ist es dem Analysten somit möglich bestimmte Transitionsverläufe (Muster) zu erkennen und für seine weitere Analyse zu benutzen.

Darstellung von Automatentransitionen Für die Analyse der Trace-Files ist eine Darstellung der Automatentransitionen essentiell. Die Darstellung muss hierbei in grafischer und textueller bzw. tabellarischer Form möglich sein.

Darstellung von Rohdaten Den Analysten und Ingenieuren von BMW ist es wichtig, jederzeit auf die Rohdaten in textueller Form zugreifen zu können, um ggf. tiefer in die Analyse gehen zu können. Dies wird benötigt, weil die Fachleute sehr vertraut mit dieser Art der Darstellung sind und die benötigten Daten direkt interpretieren können.

Präattentive Verarbeitung Damit soll erreicht werden, dass die Daten nach bestimmten Kriterien bereits kognitiv voranalysiert werden. Dies heißt insbesondere, dass bestimmte Daten auf eindeutige Weise farblich oder örtlich voneinander unterschieden werden, um den Analysten die Arbeit zu vereinfachen und vor Allem die Analyse zu beschleunigen.

Historisierung Es wird gewünscht, dass alle Analyseschritte rückgängig gemacht werden können und ggf. wiederholt werden können. Hinzu kommt, dass eine vollständige Historisierung des gesamten Analyseprozesses verfügbar sein sollte.

Filterung Dem Analysten soll eine Filterung der, in einer Trace-File enthaltenen, Automaten ermöglicht werden. Die Filterung kann hierbei nach unterschiedlichen Kriterien erfolgen. Neben der Anwendung des Filters innerhalb einer Visualisierung muss zudem die Speicherung eines Filters ermöglicht werden.

Export Dem Analysten soll es ermöglicht werden den aktuellen Analyseausschnitt in einem angemessenen Format zu exportieren.

Anforderungen an die Visualisierungen

Es werden unterschiedliche Arten von Darstellungsformen für Daten benötigt. Hierbei ist es notwendig, dass der Nutzer die Daten von abstrakt bis detailliert darstellen und in diesen navigieren kann. Das System muss hierbei einen permanenten Überblick über die gezeigten Daten anbieten.

Visualisierungsarten zur Automatenübersicht und automateninternen Sicht Zur Betrachtung der Automatenzustände, der Zustandswechsel und den Rohdaten werden folgende Visualisierungstypen angeboten:

Automatenansicht Übersicht über alle oder ausgewählte Automaten in Balkenform. Farbliche Markierungen zeigen Fehler, Warnungen usw. in den Automaten an.

Zustandssequenzdiagramm Zur Visualisierung der Zustände zuvor selektierter Automaten wird das Zustandssequenzdiagramm genutzt.

Transitionsansicht Die Transitionsansicht repräsentiert in tabellarischer Form die Transitionen aller Automaten, welche in der geladenen Trace-File enthalten sind. Hierbei werden ebenfalls farbliche Markierungen genutzt, um den Transitionstyp möglichst schnell erkennen zu können.

Rohdatenansicht Die Rohdatenansicht repräsentiert in tabellarischer Form alle Rohdaten innerhalb der geladenen Trace-File.

Visualisierungshandling Abhängig von der Visualisierung müssen Zooming (vergrößern/ verkleinern) und Panning (verschieben) innerhalb der Visualisierung unterstützt werden.

Anforderungen an die kollaborative Arbeit

Kopie des Analyseablaufs Dem Analysten soll es möglich sein den aktuellen Analysablauf oder Teile des Analyseablaufs zu kopieren und einem weiteren Analysten zur kollaborativen Arbeit zur Verfügung zu stellen.

Spiegelung des Analyseablaufs Die Anwendung soll die Spiegelung eines Analyseablaufs ermöglichen. Dies bedeutet, dass ein Analyseablauf zunächst kopiert und anschließend verknüpft wird. Die Verknüpfung sorgt dafür, dass eine Änderung in einem Analyseablauf zur gleichen Änderung im anderen Analyseablauf führt.

Vergleich von zwei Analyseabläufen Da die Workbenches frei positioniert und skaliert werden können, ist es möglich zwei Analyseabläufe oder Ausschnitte nebeneinander zu positionieren und anschließend kollaborativ von zwei Analysten zu vergleichen.

12.1.2 Nichtfunktionale Anforderungen

Zuverlässigkeit

Das System wird in einem stabilen Zustand ausgeliefert. Garantiert wird dies durch regelmäßige Tests des entwickelten Systems, welche alle Anwendungsfälle abbilden. Die Wiederherstellbarkeit wird auf Analyseebene durch eine Speicherfunktion, sowie einer Historie über die Undo- und Redo-Funktionalität zur Verfügung gestellt werden.

Design

Das Design des Systems soll den Analyseprozess optimal unterstützen und nicht von den wesentlichen Zielen der Analyse ablenken. Durch eine intelligente Anordnung der Bedienelemente soll die Benutzung des Systems erleichtert werden. Dies wird im folgenden Punkt (Benutzbarkeit) näher erläutert.

Benutzbarkeit

Die Benutzung des Systems soll sich an gängigen Standards orientieren und so möglichst einfach erlernbar sein. Die Gesten, die zur Bedienung des Systems genutzt werden, sollen einfach und intuitiv sein. Da das System für eine Gruppe von Fachspezialisten entworfen wird, ist ein gewisse Einarbeitungszeit nötig um dieses in vollem Umfang bedienen zu können. Dennoch sollen die grundlegenden Funktionen so intuitiv gestaltet werden, dass jedem ein schneller Einstieg ermöglicht wird und der volle Funktionsumfang in der Folge langsam erlernt werden kann.

Leistung und Effizienz

Um das System benutzbar zu halten, sollte es möglichst geringe Antwortzeiten vorweisen. Dies gilt insbesondere für Datenabfragen und die gestische Interaktion mit dem System. Der Ressourcenbedarf des Systems spielt eine untergeordnete Rolle. Die Wirtschaftlichkeit ergibt sich aus dem jeweiligen Analyseumfang.

Wartbarkeit und Änderbarkeit

Durch eine starke Modularisierung der Komponenten soll eine hohe Wartbarkeit und Änderbarkeit erreicht werden. Durch iteratives Vorgehen in der Entwicklung befindet sich das System stets in einem änderbaren Zustand. Eine ausführliche Dokumentation des Programmcodes soll ebenfalls zur Wartbarkeit und Verständlichkeit des Systems beitragen.

Portierbarkeit

Entwickelt wurde die Anwendung für den interaktiven Tisch vom Offis. Darüber hinaus ist die Anwendung auf allen Multi-Touch-Systemen, die Windows 7 Touch Events verwenden, lauffähig. Weiterhin ist es möglich mit Hilfe der Software Multi-Touch-Vista ein Multi-Touch-System zu emulieren und somit lässt sich die Anwendung auch mit jedem Computer, auf dem Windows 7 installiert ist, bedienen.

Sicherheit

Die Einsatzumgebung des Systems ist eine private Netzwerkumgebung, deren Sicherheit als Bedingung vorausgesetzt wird. Darüber hinaus werden die grundsätzlichen Sicherheitsmechanismen des Betriebssystems genutzt.

Korrektheit

Die Korrektheit der in unserem System implementierten Algorithmen werden durch ausführliche Tests gewährleistet. Die fehlerhafte Interpretation von Trace-Files durch einen oder mehrere Analysten kann jedoch nicht auf Softwareseite abgefangen werden.

Flexibilität

Das System kann generell alle Trace-Files, die Automatentransitionen und Rohdaten beinhalten, darstellen. Voraussetzung hierfür ist, dass die Datenquellen in einem standardisierten Format (XML) vorliegen. Es ist möglich sowohl Trace-Files, die nur aus Automatentransitionen bestehen, als auch Trace-Files, die nur aus Rohdaten bestehen, anzuzeigen.

Skalierbarkeit

Die Anwendung kann mit beliebig großen Trace-Files arbeiten. Die einzige Auswirkung einer Trace-File mit einer sehr großen Anzahl an Rohdaten ist, dass das Laden der Trace-File entsprechend länger dauert. Zudem kann der Anwender durch das Workbenchkonzept erkennen, wo die Leistungsgrenze liegt und Visualisierungen ein- und ausblenden bzw. mehr oder weniger detailliert darstellen.

12.2 Visualisierungen

Im Folgenden werden die gefundenen Visualisierungen, die den Analysten im Analyseablauf unterstützen sollen, dargestellt und beschrieben. Die Visualisierungen bauen aufeinander auf. Die erste Visualisierung ist die Automatenansicht, gefolgt vom Zustandssequenzdiagramm. Darauf folgen die Transitionsansicht sowie die Rohdatenansicht. Abschließend wird der Ablauf sowie die Ergebnisse der Evaluation zu den Visualisierungen und dem Interaktionsdesign beschrieben.

12.2.1 Beschreibung der Visualisierungen

Automatenauswahl

Die Visualisierung der Automatenauswahl bzw. -ansicht wird in Abbildung 12.1 dargestellt. In ihr werden alle Automaten angezeigt, die in einer zuvor geladenen XML-Datei enthalten sind. Durch diesen grafischen Überblick aller Automaten soll ein möglichst guter Einstieg in die Analyse unterstützt werden. Insgesamt können fünf verschiedene Farben auftreten. Diese Farben weisen auf die Zustände innerhalb eines Automaten hin und werden wie folgt unterschieden:

1. Init (grau)
2. Info (blau)
3. Okay (grün)
4. Warning (gelb)
5. Defect (rot)

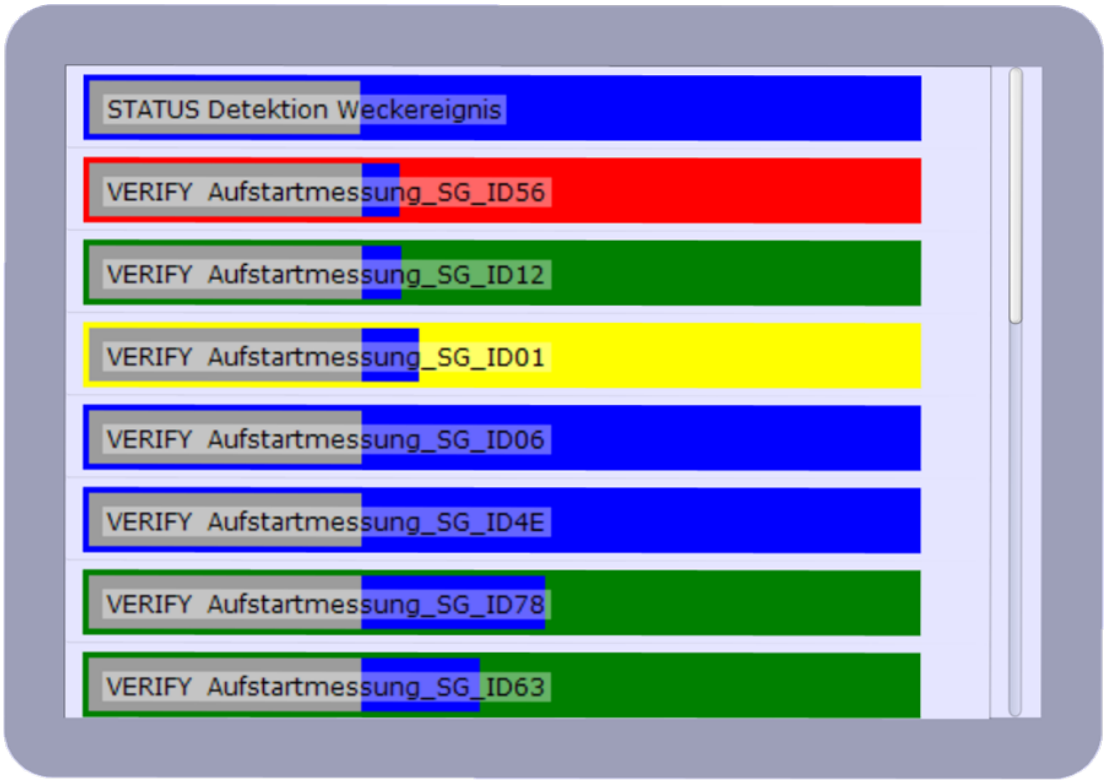


Abbildung 12.1: Darstellung der Automatenauswahl

Ein Rechteck innerhalb dieser Automatenansicht stellt genau einen Automaten dar. Dies wird in Abbildung 12.2 dargestellt. Jeder Automat besitzt einen Rahmen in einer bestimmten Farbe. Diese Farbe verdeutlicht den interessantesten bzw. bedeutendsten Zustand eines Automatens für den Analysten. Dabei nehmen die Zustände von „Init“ nach „Defect“ (der kritischste Zustand) an Bedeutung zu. In diesem Beispiel ist der Rahmen rot dargstellt und weist darauf hin, dass der Automat einen Defect enthält. Innerhalb dieses Rahmens befinden sich unterschiedliche Unterteilungen mit verschiedenen Farben und stellen den Verlauf der Zustandsübergänge dar. Die Breite der Automaten ist relativ zur Gesamtlaufzeit einer Testfahrt. Daher haben die Zustandswechsel innerhalb eines Rahmens ebenso einen zeitlichen Bezug. Durch diese Eigenschaften soll der Analyst möglichst gut bei der Vorabauswahl unterstützt werden. Wenn ein Automat daraufhin detaillierter betrachtet werden soll, muss dieser lediglich berührt werden und das Zustandssequenzdiagramm öffnet sich in einem weiteren Arbeitsbereich.

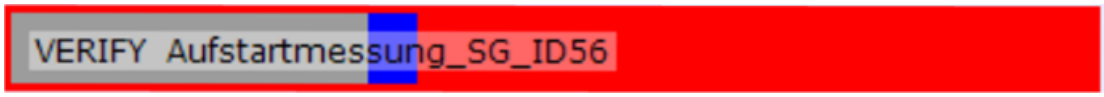


Abbildung 12.2: Darstellung eines Automaten

Zustandssequenzdiagramm

Zur detaillierteren Visualisierung eines zuvor ausgewählten Automaten wird die Darstellung in Abbildung 12.3 verwendet. Hier ist die Zustandsabfolge des Automaten auf einer Zeitachse aufgetragen. Somit werden die Zustandsübergänge mit ihren Zeitpunkten für den Analysten überschaubar visualisiert. Das Zustandssequenzdiagramm ist in einzelne Spalten gegliedert, die jeweils einen Zustand des betrachteten Automaten repräsentieren. Die Kopfzeile des Diagrammes enthält die Namen der jeweiligen Zustände. Die Zeiten in denen ein Zustand aktiv ist, werden durch einen farbigen Balken in der Spalte des jeweiligen Zustands repräsentiert. Die Farbgebung ist dabei die gleiche wie in der Automatenauswahl. Transitionen sind als Pfeile zwischen den einzelnen Balken der aktiven Zustandszeiten dargestellt. Um detailliertere Informationen zu einem Zustandswechsel zu erhalten kann ein Balken einer aktiven Zustandszeit berührt werden. Daraufhin öffnet sich die Transitionsansicht und springt zu dem Zeitpunkt an dem sich der Zustandswechsel in dem berührten Zustand ereignet. Eine Möglichkeit detailliertere Informationen direkt im Zustandssequenzdiagramm zu erhalten, ist das Zooming und Panning über die Dimension der Zeit, was beim Interaktionsdesign der Visualisierungen (siehe Abschnitt 12.4.2) näher beschrieben ist. Sehr kurze aktive Zustandszeiten, die normalerweise nicht sichtbar wären, sind durch sehr kurze Balken visualisiert, wobei die Transitionseingänge und -ausgänge mittig zu diesen sind. So kann der Analyst diese auch in der größten Übersichtsstufe erkennen, was insbesondere für sehr kurze Fehlerzustandszeiten relevant ist.

Transitionsansicht

Die Transitionsansicht wird in Abbildung 12.4 dargestellt und entspricht einer tabellarischen Darstellung. Jede Zeile entspricht dabei einer Transition, also einem Wechsel von Zustand A zu Zustand B. In der ersten Spalte wird farblich hervorgehoben, in welchen Zustand innerhalb einer Zeile übergegangen wird. Die zweite Spalte enthält den Zeitstempel einer Transition. Standardmäßig ist die Liste auch nach diesem Zeitstempel sortiert, es besteht aber auch die Möglichkeit, nach allen anderen Spalten zu sortieren. Beim Aufruf der Transitionsliste wird außerdem bereits zum passenden Zeitstempel gesprungen. Die Spalten weisen folgende Informationen auf:

1. Farbgebung des Zielzustandes

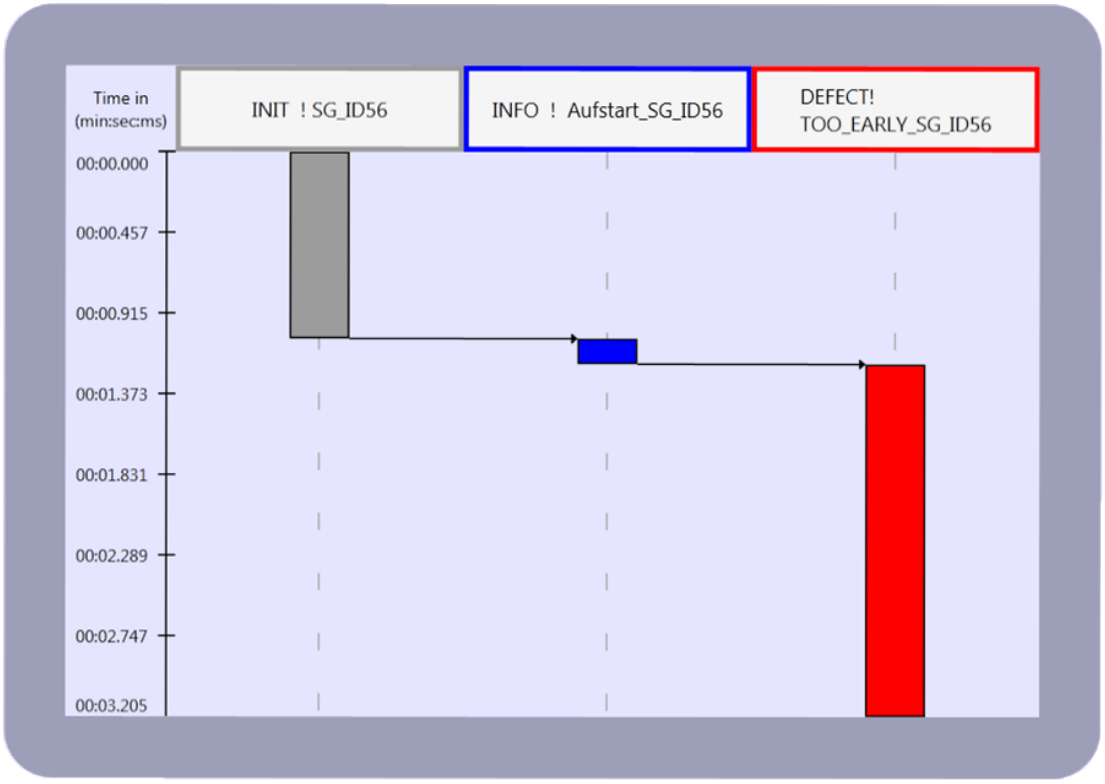


Abbildung 12.3: Darstellung des Zustandssequenzdiagrammes

2. Zeitstempel
3. Startzustand
4. Transitionsname
5. Zielzustand

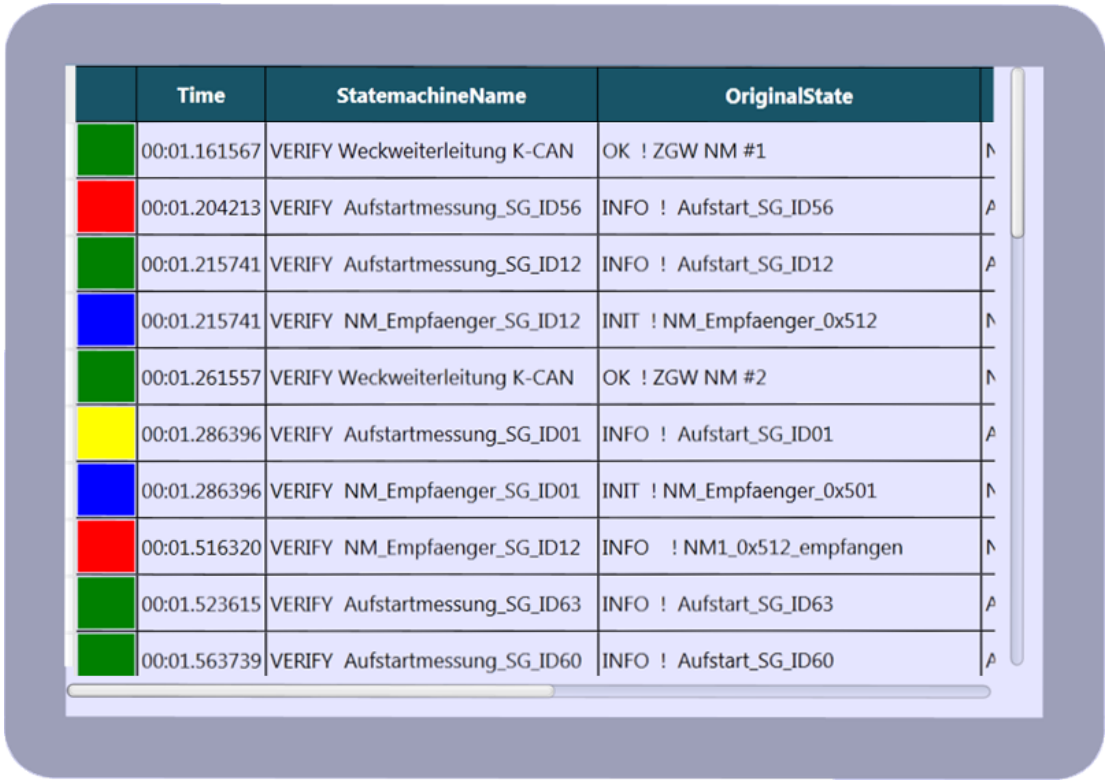


Abbildung 12.4: Darstellung der Transitionsliste

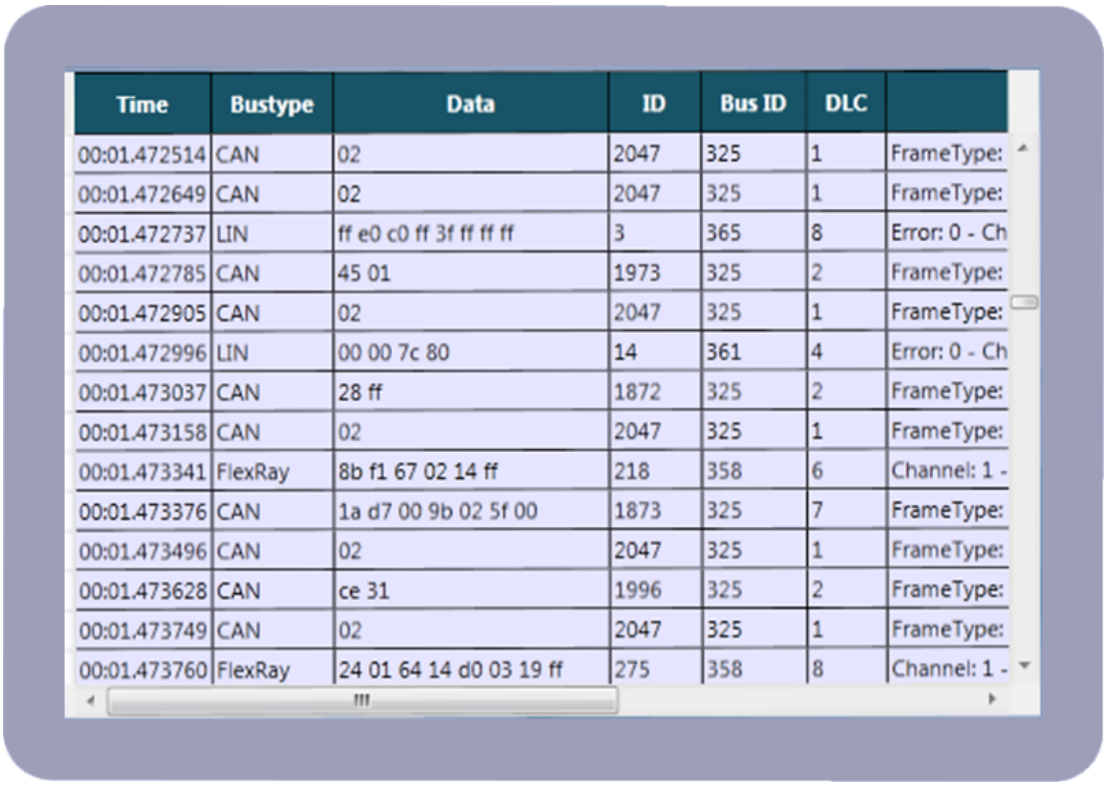
Rohdatenansicht

Die Rohdatenansicht wird in Abbildung 12.5 dargestellt. Wie ihr Name schon sagt, stellt sie die Rohdaten zur Analyse bereit. Ebenso wie die Transitionsansicht ist dies eine tabellarische Darstellung. Auch diese Liste springt anfangs zum zuvor ausgewählten Zeitstempel und ist für alle Spalten sortierbar (die Anfangssortierung geschieht nach Zeitstempel). Verschiedene Bustypes weisen unterschiedliche Parameter auf. In dieser Darstellung werden daher nur Spalten verwendet, die bei fast allen Bustypes auftreten. Diese sind:

1. Zeitstempel

2. Bustype
3. Daten
4. ID
5. Bus-ID
6. DLC

Darüber hinaus gibt es eine letzte Spalte für weitere Daten, in der alle weiteren, bussspezifischen Parameter enthalten sind.



Time	Bustype	Data	ID	Bus ID	DLC	
00:01.472514	CAN	02	2047	325	1	FrameType:
00:01.472649	CAN	02	2047	325	1	FrameType:
00:01.472737	LIN	ff e0 c0 ff 3f ff ff ff	3	365	8	Error: 0 - Ch
00:01.472785	CAN	45 01	1973	325	2	FrameType:
00:01.472905	CAN	02	2047	325	1	FrameType:
00:01.472996	LIN	00 00 7c 80	14	361	4	Error: 0 - Ch
00:01.473037	CAN	28 ff	1872	325	2	FrameType:
00:01.473158	CAN	02	2047	325	1	FrameType:
00:01.473341	FlexRay	8b f1 67 02 14 ff	218	358	6	Channel: 1 -
00:01.473376	CAN	1a d7 00 9b 02 5f 00	1873	325	7	FrameType:
00:01.473496	CAN	02	2047	325	1	FrameType:
00:01.473628	CAN	ce 31	1996	325	2	FrameType:
00:01.473749	CAN	02	2047	325	1	FrameType:
00:01.473760	FlexRay	24 01 64 14 d0 03 19 ff	275	358	8	Channel: 1 -

Abbildung 12.5: Darstellung der Rohdatenansicht

12.2.2 Evaluation

Geeignete Visualisierungsformen stellen einen wichtigen Teil in der Entwicklung einer Anwendung dar. Daher sind verschiedene Evaluationen während des gesamten Projektfortschritts durchgeführt worden.

Die erste Evaluation hatte zum Ziel, geeignete Visualisierungen sowie ein sinnvolles Interaktionsdesign zu finden. Dazu wurden verschiedene Visualisierungsvorschläge für bestimmte Bereiche (bspw. für die Synchronisation und der History) gestaltet und den Probanden vorgestellt. Weiterhin wurden die Probanden gebeten, Funktionen im System, durch die gestische Interaktionen auf den Vorschlägen, mit Ihren Händen auszulösen. Des Weiteren wurden die Probanden dazu aufgefordert, alle Aktionen, die sie durchführen wollen und auch das daraufhin erwartete Verhalten des Systems, verbal zu kommentieren. Also „laut zu denken“, was der Evaluationsmethode des „Thinking-Aloud“ entspricht. Währenddessen wurden die Interaktionen dokumentiert, die auf dem Vorschlägen durchgeführt wurden, um sie zur späteren Auswertung zu nutzen. In diesem Kontext wurden außerdem die Aktionen während der Studie gefilmt, damit bei der Evaluationsauswertung keine wichtigen Details vergessen werden.

Aufbau

Im diesem Kapitel sollen der Aufbau sowie alle verwendeten Mittel der Evaluation beschrieben werden.

Hardware: In der Abbildung 12.6 wird der Aufbau der Studie dargestellt, der im wesentlichen aus zwei Gegenständen bestand. Zum Einen bestand der Aufbau aus einer Digitalkamera (1), welche auf einem Stativ befestigt wurde, um alle Interaktionen der Probanden aufzuzeichnen. Diese Art der Dokumentation diente zur späteren, detaillierteren Auswertung. Zum Anderen wurde der Multitouch Tisch (2) genutzt, auf dem der Prototyp ausgeführt wurde und die Probanden interagiert haben.

Software: Damit die Studie auf einer möglichst weit fortgeschrittenen Anwendung ausgeführt wird, ist ein grafischer Prototyp verwendet worden. Dieser Prototyp setzte sich aus verschiedenen Microsoft PowerPoint Folien zusammen, auf denen einzelne Programmzustände grafisch dargestellt wurden. Auf diesen „Programmzuständen“ sind die Interaktionen durchgeführt worden.

Dokumente: Es wurden verschiedene Dokumente für die Studie angefertigt. Zum Einen ein Informationsblatt für die Teilnehmer. Dieses Schriftstück erklärt kurz die Hintergründe der Studie, erläutert was die Probanden im Falle einer Teilnahme erwartet und informiert sie über deren Rechte als Studienteilnehmer. Außerdem wurde ein Dokument zur Aufnahme von demographischen Daten angefertigt. Auf diese Weise erhielten wir Auskunft über das Geschlecht, das Alter sowie die Multitoucherfahrung der Probanden. Zudem wurde ein Schriftstück zur Dokumentation der durchgeführten Gesten und der präferierten Designvorschläge ausgearbeitet, um die Beobachtungen unmittelbar nieder

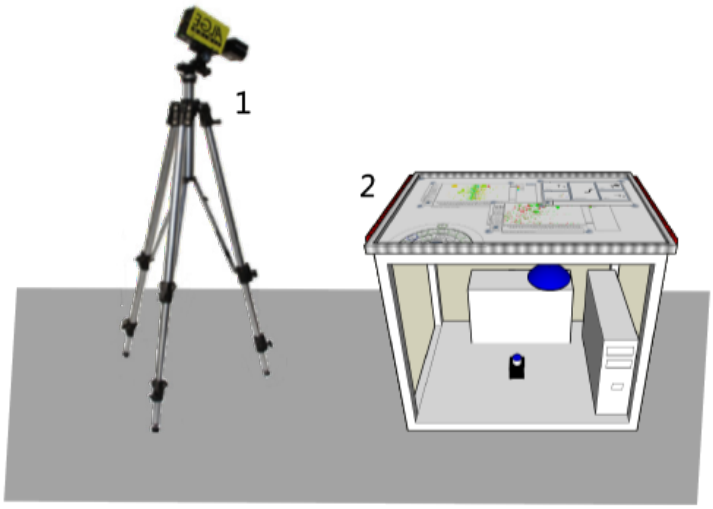


Abbildung 12.6: Evaluationsaufbau

zuschreiben. Abschließend wurden weitere Dokumente erstellt, die den geregelten Ablauf der Evaluation garantierten. Beispielsweise ein Dokument welches den Ablauf der Evaluation beschrieb, ein Dokument welches die einzelnen Zeitslots beinhaltete sowie ein Excel-Sheet zur schnelleren Auswertung der demographischen Daten.

Informationen zu den Probanden

Insgesamt wurde die Evaluation mit zehn Probanden durchgeführt. Davon waren sieben männlichen und drei weiblichen Geschlechts, wodurch ein guter Durchschnitt erzielt wurde. Bei der Einladung der Probanden wurde darauf geachtet, nicht nur Personen aus dem Informatikbereich oder Personen die mit Vorwissen im Multitouchbereich ausgestattet sind, für uns zu gewinnen. Die Probanden waren hauptsächlich Studierende aus verschiedenen Studiengängen sowie Mitarbeiterinnen und Mitarbeiter des OFFIS.

Dem dargestellten Diagramm (vgl. Abbbildung 12.7) kann entnommen werden, dass der Großteil (60%) der Teilnehmer im Altersbereich von 21-27 Jahre lag. Auf einer Skala von 1 (keine Erfahrung) bis 5 (viel Erfahrung) haben die Probanden sich selbst durchschnittlich auf eine Multitoucherfahrung von 2,8 eingeschätzt (Standardabweichung beträgt ≈ 1.48).

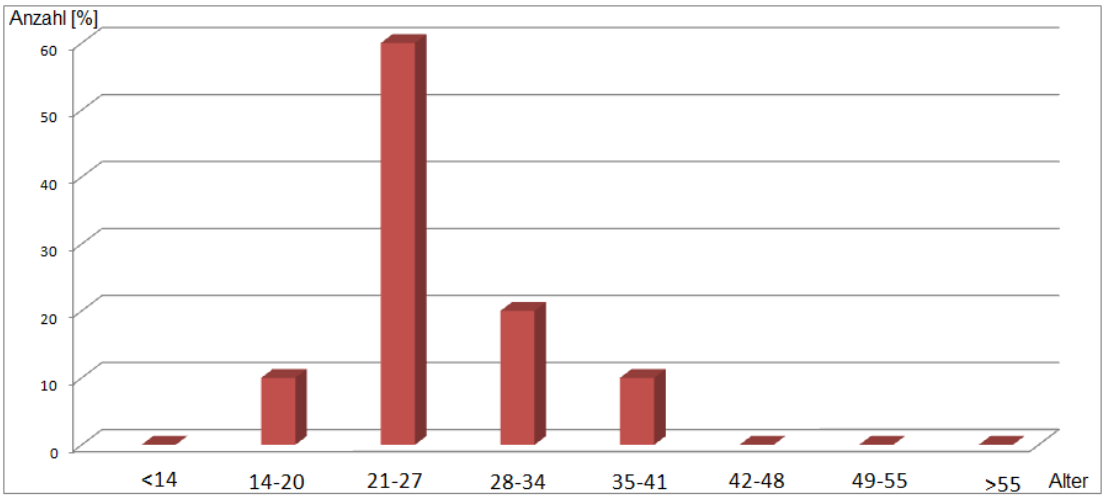


Abbildung 12.7: Alter der Probanden

Ablauf der Evaluation

Zu Beginn wurde in der Einladung zur Evaluation ein Doodle-Link angefügt. dort konnten die Probanden ein Zeitslot, an dem sie teilnehmen wollten, auswählen. Diese Zeitabschnitte waren mit 30 Minuten pro Teilnehmer kalkuliert. Im Durchschnitt hat eine Evaluation etwa 22 Minuten in Anspruch genommen. Daher bekamen die Probanden eine kleine Aufwandsentschädigung zum Schluss der Evaluation.

Die eigentliche Evaluation bestand aus vier wesentlichen Teilen. Als Erstes haben die Probanden die Dokumente (demographische Daten, Einverständniserklärung) ausgefüllt. Danach wurden den Probanden wesentliche Dinge zum Verständnis der Problematik dargestellt. Dazu wurden zwei Powerpoint-Folien erstellt. Hier wurden hauptsächlich die Fragen „Welchen Nutzen wird die Anwendung später haben?“ und „Welche Zusammenhänge haben die zukünftigen Visualisierungen?“ beantwortet. Diese Erklärungen sind äußerst wichtig gewesen, um den Probanden eine Vorstellung der Zusammenhänge des Programms zu vermitteln. Im dritten Teil der Studie wurden den Probanden anhand des Powerpoint-Prototyps verschiedene Ausgangszustände dargelegt und durch eine darauf ausgeführte Interaktion, resultierende Zielzustände des Programms angezeigt. Die Aufgabe der Teilnehmer bestand darin sich eine Interaktion auszudenken, um den Zielzustand zu erreichen. Das weitere Vorgehen der Interaktionsevaluation wird in dem Kapitel 12.4.4 beschrieben.

Den letzten Teil der Evaluation stellten die verschiedenen Design-Vorschläge dar, die den Probanden vorgestellt wurden. Auf diese soll an dieser Stelle näher eingegangen werden. Die Aufgabe der Probanden war in diesem Fall, einen Vorschlag zu präferieren oder eigene Design-Ideen hervorzubringen.

Es wurden zu folgenden Programmzuständen Design-Vorschläge evaluiert:

- Auswahl der Automaten
- Das Workbenchhandling
- Verbindung zweier Workbenches
- Einige Menü Beispiele
- Einblenden der History

Ergebnisse

Im Folgenden werden die bedeutendsten Ergebnisse der Designvorschläge vorgestellt.

- **Auswahl der Automaten:** Bei der Automatenauswahl handelte es um die Selektierung einzelner Automaten und darauf folgend den Wechsel in das Zustandssequenzdiagramm. Zu diesem Anwendungsfall haben wir vier Vorschläge erarbeitet, wobei die Probanden sich nicht auf einen Vorschlag festlegen konnten. Vorschlag 1 bestand daraus, durch einen Button, der die jeweils andere Ansicht symbolisiert, zwischen den beiden Visualisierung schnell wechseln zu können. Dieser Vorschlag wurde fünf Mal bevorzugt. In Vorschlag 2, der drei Erststimmen und zwei Zweitstimmen bekam, wurde durch Tabs zwischen den beiden Visualisierungen gewechselt. Das selbe Ergebnis erzielte Vorschlag 4, wo zwei Workbenches miteinander verbunden waren und in der einen Workbench die Automatenansicht und in der zweiten das Zustandssequenzdiagramm visualisiert wird. Der Vorschlag 3, bei dem ein Popup-Fenster über den Rand ausfahrbar war, wurde von einem Teilnehmer nur einmal als Zweitstimme gewählt und erhielt daher bei der späteren Designentscheidung keine große Beachtung. Durch die gute Bewertung des vierten Vorschlags und der mehrheitlichen Meinung innerhalb der Projektgruppe entschieden wir uns den vierten Vorschlag weiter nachzugehen. Dieser ergab den positiven Effekt, dass über eine Workbench mehrere Automaten jederzeit direkt an und abwählbar sind und daher mehrere Zustandssequenzdiagramme geöffnet bzw. geschlossen werden können.
- **Workbenchhandling:** Bei diesen Anwendungsfall handelte es sich darum, die Workbench am intuitivsten zu verschieben, rotieren oder die Größe zu bestimmen. Dafür hatten wir uns wiederum vier Designvorschläge überlegt, aus denen allerdings nur zwei Vorschläge von den Probanden gewählt wurden. Zum einen der Vorschlag 1, in dem ein breiterer Rand zum Workbenchhandling genutzt werden sollte. Dieser Vorschlag wurde mit sieben Erststimmen und zwei Zweitstimmen gewählt. Ein weiterer Vorschlag sah vor, dass an den Ecken einer Workbench Interaktionspunkte sitzen, auf denen das Workbenchhandling durchgeführt wird.

Dieser Vorschlag wurde mit vier Erststimmen bewertet. Die anderen Vorschläge waren zu vernachlässigen. Aufgrund der großen positiven Resonanz für den ersten Vorschlag, wurde dieser Vorschlag bei der Designentscheidung weiter verfolgt und letztendlich in die Anwendung eingebaut.

- **Verbindung zweier Workbenches:** Zur Synchronisation zweier Workbenches wurden sechs Designs evaluiert, wobei zwei Darstellungen hauptsächlich präferiert wurden. Zum Einen der Vorschlag 6, wo eine Verbindung zwischen zwei Workbenches mit dem Finger gezogen wird. Und sich daraufhin zwischen den Workbenches ein Menü öffnet, wo die Interaktionsoption gewählt werden kann (6 Erststimmen). Des Weiteren wurde Vorschlag 1 mit zwei Erststimmen und zwei Zweitstimmen gewählt. Hier wurde ein Button zur Verbindungsherstellung implementiert, der durch Berührung ein Menü öffnet in dem die Workbench ausgewählt werden kann, mit der man interagieren möchte sowie die Interaktionsoptionen. Nachdem diese Schritte abgeschlossen wurden, wird durch Einfärbung des Buttons signalisiert, welche Workbenches miteinander interagieren. Die anderen Vorschläge sind zu vernachlässigen. Die daraus resultierten Ergebnisse wurden genauer analysiert und zum Teil in die Anwendung eingebaut. So wird unter anderem eine Verbindungslinie zwischen zwei zusammengehörenden Workbenches angezeigt, aber auch der Rand einer Workbench wird farblich hervorgehoben, sofern diese im Synchronisationsmenü ausgewählt wurde.
- **Menü Beispiele:** Von den fünf Designvorschlägen zu einen geeigneten Menü, gab es drei, die besonders gut abgeschnitten hatten. Am besten beurteilt wurde der Vorschlag 3 mit vier Erststimmen. Hier ist das Menü längs unter der Workbench angeordnet und kann je nach Bedarf geöffnet oder geschlossen werden. Danach wurden die Vorschläge 1 und 2 mit jeweils drei Erststimmen und einer Zweitstimme priorisiert. Im Vorschlag 1 wird ein Markup-Menü dargestellt. Der Aufbau eines Markup-Menüs ist durch einen zentralen Button definiert, von dem aus hierarchisch weitere Menüpunkte aufgerufen werden können. Durch die Move-Geste kann in der Hierarchie die gewünschte Option gewählt werden. Beim Vorschlag 2 wird ein Halfpie-Menü abgebildet. Die Form eines Halfpie-Menüs ist ein Halbkreis. Durch das Interagieren auf diesem Halbkreis öffnet sich eine weitere Schicht oberhalb dieses Halbkreises, wo verschiedene Menüpunkte zur Verfügung stehen. Diese öffnen durch Betätigung wiederum eine weitere Schicht mit Menüpunkten. Nach längeren Diskussionen entschiedenen wir uns für ein kontextunabhängiges Marking-Menü, welches an jeglicher Stelle innerhalb einer Workbench aufgerufen werden kann.
- **Einblenden der History:** Durch diesen Anwendungsfall wollten wir das Ergebnis erzielen, zu wissen, ob der Proband die History durch das Ausführen einer Geste oder über ein Menü öffnen würde. Das Ergebnis ergab, dass sieben Erststimmen der Probanden das Aufrufen der History über einem Menü bevorzugen. Der zweite

Vorschlag bekam fünf Erststimmen und eine Zweitstimme. Folglich wurden zwölf Erststimmen vergeben, die aus der Unentschlossenheit der Teilnehmer resultieren. Jedoch schien die Mehrzahl für die Bedienung über ein Menü zu sein. Diesen Vorgang verfolgten wir somit auch in unserem weiteren Projektverlauf. Außerdem werden dadurch Fehleingaben verhindert, die durch eine gestische Bedienung entstehen könnten.

12.3 Funktionen

12.3.1 Allgemeiner Analyseablauf

Der grobe Ablauf des Programms gestaltet sich derart, dass der Anwender zunächst Daten eines Testlaufs mit einem Fahrzeug in das Programm lädt, um diese Daten, grafisch aufbereitet, analysieren zu können. Als Eingabeformat akzeptiert TOAD XML-Dateien mit optional integrierten Rohdaten. Es ist dabei möglich, in einer Programminstanz mehrere verschiedene Datensätze gleichzeitig zu analysieren. Die gleichzeitige Analyse kann wahlweise von einer oder mehreren Personen vorgenommen werden.
Der allgemeine Analyseablauf eines einzelnen Analysten gestaltet sich hierarchisch. Nach dem Laden einer Trace-File in der Datenauswahl wechselt die Visualisierung zur Automatenauswahl. Für jeden der in der Trace-File enthaltenen Automaten kann ein einzelnen Zustandssequenzdiagramm geöffnet werden. Durch Berühren eines beliebigen Zustands im Zustandssequenzdiagramm wird die Transitionsansicht erreicht. Es ist möglich beliebig viele, je nach Anzahl Zuständen im Zustandssequenzdiagramm, Transitionsansichten zu öffnen. Nun kann der Anwender durch Auswahl einer Transition die Rohdatenansicht aufrufen. Diese öffnet sich ebenfalls in einer separaten Workbench. Sollte für die gewählte Transitionsansicht bereits eine Rohdatenansicht geöffnet sein, so wird keine zweite Rohdatenansicht geöffnet, sondern die bereits geöffnete Rohdatenansicht springt zum Zeitpunkt der zuvor gewählten Transition.

12.3.2 Marking-Menü

Die entwickelte Software beinhaltet ein integriertes Marking-Menü mit dessen Hilfe alle zusätzlichen Funktionen aufgerufen werden können. Der Aufruf und die Bedienung des Menüs wird im nächstfolgendem Kapitel erläutert. Der Menüaufbau ist in vier Teile gegliedert.

History Im History-Menü befinden sich alle Funktionalitäten der History. Zur Auswahl stehen hier *Undo*, um einen Schritt rückgängig zu machen, *Redo*, um einen Schritt zu wiederholen, und *Öffnen*, um die History anzuzeigen (s. auch 12.3.3 History).

Programm Hier kann das Programm durch *Schließen* beendet werden.

Workbench Unabhängig zur Visualisierung kann mit *Schließen* die entsprechende Workbench, sowie alle nachfolgenden, geschlossen werden und über den Punkt *Synchronisation* kann das Synchronisationsmenü aufgerufen werden (s. auch 12.3.4 Synchronisation). Zudem existieren visualisierungsabhängig weitere Funktionen, welche hier aufgerufen werden können. Mit der Funktion *Filter setzen* (nur im Zustandssequenzdiagramm) können Filterkriterien gesetzt werden und mit der Funktion *Filter anwenden* (nur in der Automatenauswahl) kann bestehende Automatenliste nach einem zuvor gesetztem Filter gefiltert werden (s. auch 12.3.5 Filter). Durch die Funktion *CSV Export* (nur Transitionsansicht und Rohdatenansicht) können die aktuell angezeigten Listen im CSV-Format exportiert werden (s. auch 12.3.6 Export).

Außerdem enthält jeder Menüpunkt den Eintrag *Zurück*, mit Hilfe dessen in die darüberliegende Menüebene zurückgekehrt werden kann.

12.3.3 History

Durch Aufruf der History (Menü → History → Öffnen) erscheint unterhalb der entsprechenden Workbench die History. Darin sind der aktuelle Zustand der Sitzung, sowie jeweils zwei Schritte davor und danach (falls vorhanden) zu sehen. Der Analyst kann mit Hilfe der History im Analyseablauf vor und zurück navigieren. Hierbei ist es möglich einen bestimmten Zustand direkt anzuwählen oder jeweils einen Schritt vor oder zurück zu navigieren. Zudem kann der Analyst mit Hilfe der History den aktuellen Analyseablauf speichern. Das Speichern des aktuellen Analyseablaufs wird mit Hilfe der Schaltfläche *Session speichern* realisiert.

12.3.4 Synchronisation

Durch die Synchronisation kann der Anwender die aktuelle Workbench mit einer anderen verbinden. Dazu muss mindestens eine Workbench im Datenauswahlmodus vorhanden sein. Es gibt zwei Möglichkeiten, zwei Workbenches miteinander zu verbinden.
Zum Einen kann der Anwender eine Einmalkopie anfertigen. Dabei wird der Inhalt der aktuellen Workbench auf die andere kopiert. Dieser Vorgang kann wahlweise mit oder ohne allen nachfolgenden Schritten durchgeführt werden. Nach Abschluss der

Kopie sind beide Workbenches unabhängig voneinander und können getrennt verwendet werden.
Zum Anderen können die Workbenches direkt miteinander verknüpft werden. Das bewirkt, dass beide Workbenches synchron zueinander sind. Wenn eine Aktion auf einer der beiden Workbenches ausgeführt wird, wird sie synchron auf der anderen ausgeführt.

12.3.5 Filter

Mit Hilfe des Filters ist es dem Anwender möglich in der Automatenansicht nur die für ihn interessanten Automaten anzeigen zu lassen. Die Filterung geschieht also immer auf Automatenebene und grenzt die Anzahl der anzuzeigenden Automaten ein.

Das Setzen eines Filters geschieht im Zustandssequenzdiagramm. Mit Hilfe des Menüs kann der Anwender von Zustandssequenzdiagramm in die Ansicht zum Setzen eines Filters gelangen. Der zu setzende Filter bezieht sich immer auf den zuvor im Zustandssequenzdiagramm angezeigten Automaten. Es kann nach folgenden Kriterien gefiltert werden:

- Automatenname: In der Automatenansicht werden nur diejenigen Automaten angezeigt, welche den gleichen Automatennamen haben.
- Zustandsabfolge: In der Automatenansicht werden nur diejenigen Automaten angezeigt, welche eine identische Zustandsfolge (z.B. von INIT über OKAY zu DEFECT) wie die gespeicherte Zustandsfolge enthalten.
- Zustände: In der Automatenansicht werden nur diejenigen Automaten angezeigt, welche einen identischen Zustand wie den gespeicherten Zustand enthalten.
- Zustandsübergänge (Transition): In der Automatenansicht werden nur diejenigen Automaten angezeigt, welche einen identischen Zustandsübergang wie den gespeicherten Zustandsübergang enthalten.

Nachdem ein Filter gesetzt wurde, kann er in der Automatenansicht angewendet werden. Die Anwendung geschieht ebenfalls mit Hilfe des Menüs. Durch die Anwendung eines Filters wird die Automatenansicht nach den im Filter gesetzten Kriterien gefiltert.

12.3.6 Export

Durch diese Option kann der Anwender die aktuellen Inhalte der Transitionsansicht oder der Rohdatenansicht exportieren. Da es sich bei beiden Visualisierungen um tabellenartige Ansichten handelt, wurde als Exportformat CSV gewählt. Nachdem die aktuelle Ansicht in eine CSV-Datei exportiert wurde, kann der Analyst den Export mit jedem beliebigen Texteditor öffnen und weiter analysieren.

12.4 Interaktionsdesign

Dieser Abschnitt beschreibt, wie das Interaktionsdesign des Prototypen umgesetzt wurde. Im Rahmen dessen werden das Workbenchhandling, die Interaktion in den Visualisierungen sowie die Menüinteraktion beschrieben. Abschließend werden im letzten Abschnitt einige Ergebnisse der ersten Evaluation vorgestellt.

12.4.1 Workbenchhandling

Workbenches (Abk. „WB“) sind die zentralen, übergeordneten Objekte im beschriebenen System. Die Workbenches enthalten die einzelnen, interaktiv änderbaren Visualisierungen. Typischerweise wird ein Analyst mit mehreren Workbenches arbeiten.
Diese Workbenches, genau wie die beinhalteten Visualisierungen, werden ausschließlich gestisch bedient. Im Folgenden sollen die Gesten zur Manipulation dieser Objekte beschrieben werden. Wissenschaftlich beschrieben sind sie zum Beispiel bei Wobrock et al.: „User-Defined Gestures for Surface Computing“. In Smartphones mit Touchscreens werden viele der verwendeten Gesten genutzt. Zusätzlich wurden diese Gesten bereits in einem frühen Stadium des Projektes evaluiert.

Erstellen der Workbench Workbenches können vom Analyst erstellt werden, indem eine einfache Linie auf dem Tisch (nicht aber beginnend in einer existierenden Workbench) gezogen wird. Dabei ist zu beachten, dass die Nutzer an unterschiedlichen Stellen des Tisches stehen können, es daher keine standardisierte Ausrichtung gibt. Die Orientierung der neuen Workbench bestimmt sich durch den Anfangs- und Endpunkt der Linie, wobei der Anfangspunkt die Unterseite der Workbench markiert, und umgekehrt. Typischerweise zieht der Analyst eine Linie von der Tischkante zur Mitte des Tisches. Die zweite aus dieser Geste genutzte Information ist die Länge des Striches. Sie entspricht der Höhe der Workbench.

Workbench verschieben Das Verschieben der Workbench lässt sich durch eine einfache Geste erreichen: Der Benutzer drückt mit einen Finger auf den Rand der Workbench und zieht diese in die gewünschte Richtung. Diese Geste ist in Abbildung 12.8 dargestellt.

Workbench rotieren Die Rotation einer Workbench kann, wie oben beschrieben, durch eine zwei-Finger-Geste erreicht werden, bei der der Abstand der Finger gleich bleibt. Lediglich die Position der beiden Finger oder eines der beiden Finger wird geändert. So kann ein Finger beispielsweise auf dem linken Rand und ein anderer auf dem rechten Rand gehalten werden und durch die anschließende Bewegung, in unterschiedlichen Richtungen, kann die Workbench rotiert werden. Auch mit dieser Geste kann durch verschieben der Berührungspunkte die Workbench verschoben werden.

Abbildung 12.8: Drag-/Move-Geste. [WMW09a]

Größe der Workbench ändern Die Größe einer Workbench lässt sich mit Hilfe der so genannten „Spread and Pinch“-Geste verändern. Als Spread-, also Spreizgeste, bezeichnet man das Auseinanderziehen zweier gedrückter Finger, intuitiv wird damit die Workbench vergrößert. Das Verkleinern funktioniert ähnlich: Durch die Pinch-, also Kneifgeste, bei der zwei in der Workbench gedrückte Finger zusammengezogen werden. Während diese Geste ausgeführt wird, können auch gleichzeitig die beiden Finger, bzw. die Workbench verschoben werden. So kann mittels einer Geste die Größe, Position und Orientierung der Workbench verändert werden. Diese Geste ist in Abbildung 12.9 dargestellt.

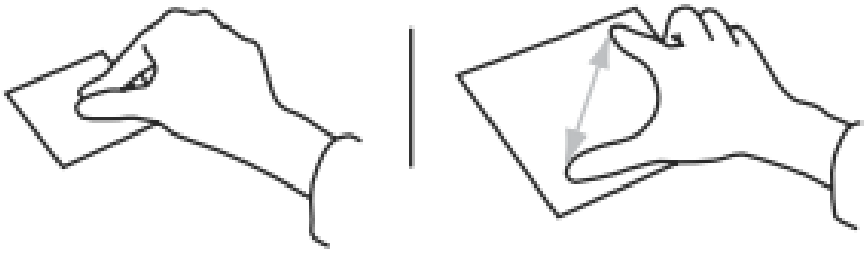


Abbildung 12.9: Spread-/Pinch-Geste. [WMW09a]

12.4.2 Visualisierungsinteraktion

Wie auch in den anderen Abschnitten wird hier lediglich die Interaktion innerhalb der Visualisierungen beschrieben. Der Aufbau sowie weitere Informationen zu den

Visualisierungen sind bereits in dem Kapitel 12.2 beschrieben worden.

Datenauswahl Nach dem Öffnen einer Workbench gelangt der Anwender zur Auswahl der Datenquelle. Dort gibt es zwei Alternativen eine Datei zu öffnen, die durch entsprechende Schaltflächen gekennzeichnet sind.

1. **Laden einer XML-Datei** Mit dieser Alternative können Daten aus einer XML-Datei geladen werden. Durch einen SingleTap wird die Schaltfläche aktiviert, anschließend erscheint ein Bereich in dem die XML-Datei durch einen weiteren SingleTap ausgewählt werden kann. Durch Berühren der zu ladenden Datei wird die entsprechende XML-Datei in das System geladen.

2. **Laden einer gespeicherten Sitzung** Durch die zweite Alternative ist es möglich, begonnene Sitzungen wieder herzustellen. Dies bietet sich insbesondere dann an, wenn die Analyse eines Problems noch nicht vollständig abgeschlossen wurde. Hierzu wird die Schaltfläche „Session laden“ durch einen SingleTap aufgerufen, wodurch der Bereich geöffnet wird, indem die entsprechende Sitzung ausgewählt und durch das Ausführen eines SingleTaps daraufhin aufgerufen werden kann.

Automatenauswahl Nachdem die Daten durch eine der zwei Alternativen in das System geladen wurden, erscheint die Automatenansicht. Hierbei sind alle in dem Datensatz enthaltenen Automaten zu sehen und entsprechend der Abbildung 12.1 gekennzeichnet. Da es sich hierbei in der Regel um eine große Menge von Automaten handelt, existiert an der rechten Seite eine Scrollbar, durch die innerhalb der Automatenansicht gescrollt werden kann. Dazu muss die Scrollbar mit einem Finger gehalten werden und kann anschließend durch das Ziehen nach unten oder oben bewegt werden. Sofern der Analyst einen Automaten auswählen will, muss dieser Automat lediglich mittels eines SingleTaps berührt werden. Dadurch wird der Automat selektiert und die Anwendung ruft automatisch das entsprechende Zustandssequenzdiagramm in einer neuen Workbench auf.

Zustandssequenzdiagramm Über das Zustandssequenzdiagramm kann die Transitionsliste aufgerufen werden. Damit diese, in einer neuen Workbench, angezeigt wird, muss ein SingleTap auf einen der Zustände ausgeführt werden, daraufhin öffnet sich automatisch die Transitionsliste. Weiterhin kann innerhalb des Zustandssequenzdiagramms gezoomt und gescrollt werden, dadurch wird dem Analyst die Möglichkeit geboten einen besseren Überblick über den Ablauf zu bekommen. Um in der Visualisierungen zu Zoomen kann eine „Spread and Pinch“-Geste durchgeführt werden. Zum Scrollen muss lediglich die Move-Geste innerhalb des Zustandssequenzdiagramms angewendet werden.

Transitionsliste Die Navigation in der Transitionsliste geschieht durch die „Move“-Geste innerhalb der Liste oder durch die Benutzung der Scrollleiste. Weiterhin

kann die Liste über die Spaltenüberschriften neu sortiert und auch neu angeordnet werden. Um eine neue Sortierung vorzunehmen muss in der Zeile der Spaltenüberschriften ein Eintrag mittels SingelTap ausgewählt werden, daraufhin wird die Liste neu sortiert. Damit die Spaltenreihenfolge neu angeordnet wird, muss ein Eintrag der Spaltenüberschriften berührt, festgehalten und anschließend an die richtige Position gezogen werden. Außerdem wird über die Transitionsliste die Rohdatenansicht aufgerufen. Dazu muss eine Zeile in der Liste mit einem SingleTap berührt werden, anschließend wird dieser Eintrag farblich unterlegt. Daraufhin wird die Rohdatenansicht erstellt, wobei der Vorgang, aufgrund der Anzahl der vorhandenen Einträge, etwas länger dauern kann. Ebenfalls wird innerhalb der Rohdatenansicht automatisch zum ausgewählten Zeitpunkt gesprungen.

Rohdatenansicht Innerhalb dieser Visualisierung werden die Rohdaten angezeigt. Da hier nicht eindeutig bestimmt werden kann, welcher Eintrag der Rohdaten zu welchem Automaten, bzw. welcher Transition gehört, wird hier ein gewisses Zeitfenster angezeigt, welches um das gewählte Datum herum angeordnet ist. In dieser Ansicht kann durch „Move“-Geste gescrollt werden, zusätzlich enthält auch diese Visualisierung eine Scrollbar. Ferner können auch in dieser Visualisierung die Spalten über die Spaltenüberschriften sortiert und neu angeordnet werden. Der Ablauf ist dabei der gleiche wie bei der Transitionsliste. Sobald ein Eintrag in der Liste mittels eines SingleTap berührt wird, wird dieser farblich gelb hervorgehoben.

Synchronisation Um eine Synchronisation zwischen zwei Workbenchen zu starten, muss zuvor im Menü der Eintrag „Synchronisation“ ausgewählt werden. Der anschließend angezeigte Bereich beinhaltet die drei Funktionen „Kopieren“, „Verknüpfen“ und „Abbrechen“. Diese Funktionen können, mittels einen SingleTaps auf diesen, ausgeführt werden. Allerdings muss bei den ersten beiden Funktionen zuerst eine ungenutzte Workbench existieren und diese anschließend in der dargestellten Liste ausgewählt werden. Dadurch erhält die entsprechende Workbench einen roten Rahmen und kann für die beiden Synchronisationsfunktionen genutzt werden.

Filter Die Filterfunktion „Filter setzen“ wird ebenso, wie die Synchronisation, über einen Eintrag im Menü, innerhalb des Zustandssequenzdiagramms, aufgerufen. Sobald dieser Eintrag ausgewählt wurde, erscheint eine Liste mit möglichen Filtern. Mittels eines SingleTaps kann der entsprechende Filter ausgewählt werden, der daraufhin in der Automatenansicht über den Menüeintrag „Filter anwenden“ angewendet werden kann. Weiterhin werden die Funktionen „Abbruch“ und „Zurücksetzen“ angeboten, damit die Aktion abgebrochen werden kann oder alle in der Automatenansicht ausgewählten Filter zurückgesetzt werden können.

History Auch die History wird über einen Menüeintrag (History → Öffnen) aufgerufen. Hierbei blendet die History sich entsprechend am unteren Rand der Workbench ein. Über die dargestellten Bilder können in der History ein oder zwei Schritte vor- oder zurücknavigiert werden. Dazu muss ein SingleTap auf den Bildern ausgeführt werden. Weiterhin können über die Pfeile an der linken und rechten Seite mehrere Schritte vor- oder zurücknavigiert werden. Um die History zu schließen muss ebenfalls der entsprechende Eintrag im Menü (History → Schließen) ausgewählt werden.

12.4.3 Die Bedienung des Menüs

Das Menü stellt einen wichtigen Aspekt innerhalb der Anwendung dar. Denn nur über das Menü können bestimmte Funktionen aufgerufen werden.

Aufruf des Menüs Damit das Menü aufgerufen wird, muss eine „Tap-and-Hold"-Geste für eine Dauer von 500ms, auf dem Rahmen einer Workbench, durchgeführt werden. Nach dem Ablauf der Zeit erscheint um den Finger ein Menü, welches die verschiedenen Funktionen beinhaltet.

Interaktion innerhalb des Menüs Um einen Menüeintrag auszuwählen darf weiterhin der Finger nicht von der Tischoberfläche abgesetzt werden. Navigiert wird hierbei mit einem Finger. Dazu wird der Finger auf den entsprechenden Menüeintrag gezogen, ohne den Kontakt zur Fläche zu lösen. Sobald ein Menüeintrag erreicht wird, leuchtet dieser rot auf. Daraufhin öffnet sich entweder die entsprechende Funktion oder eine weitere Menüebene. Über die Funktion „Zurück" gelangt der Analyst wieder eine Menüebene höher.

Schließen des Menüs Um das Menü zu schließen muss der Finger von dem Tisch abgesetzt werden. Im Anschluss daran verschwindet das Menü wieder.

12.4.4 Evaluationsergebnisse

Im Folgenden werden die Ergebnisse von der langen Nacht der Wissenschaft sowie die Ergebnisse der ersten Evaluation vorgestellt.

Vorstellung auf der langen Nacht der Wissenschaft

Bereits vor der ersten durchgeführten Evaluation wurde unsere Anwendung auf der langen Nacht der Wissenschaft, an der Universität Oldenburg, vorgestellt. Diese fand am 24. September 2010 statt [Old10]. Das Ziel der langen Wissenschaft war für uns,

unseren frühen Entwicklungsstand zu präsentieren sowie Rückschlüsse über die Qualität des Workbenchhandlings zu erlangen. Dazu haben wir aus den von uns realisierten Workbenches ein Puzzlespiel erstellt, welches die Besucher selbständig vollenden konnten. Dabei war unter anderem ersichtlich, dass viele Besucher mit ihren Ärmeln oder anderen Gegenständen auf dem Multi-Touch Tisch gelangten, das zu einigen Fehleingaben führte. Zudem haben die Teilnehmer nach den einzelnen Tests einen Fragebogen ausgefüllt. Dieser ergab, dass das System bereits zu dem frühen Zeitpunkt als gut bewertet wurde (drei bis vier, von fünf Punkten). Der Fragebogen befasste sich unter Anderem mit folgenden Fragestellungen: ob das Erstellen eines Puzzleteils einfach war, ob das Anordnen der Puzzleteile einfach war oder ob die Zusammenarbeit mit anderen Besuchern an dem Tisch gut funktioniert hatte.

Ergebnisse der ersten Evaluation

Wie das Kapitel 12.2.2 beschreibt, wurde bereits in der ersten Evaluation Teilaspekte des Interaktionsdesigns evaluiert. Diesbezüglich hatten die Probanden die Aufgabe, sich eine Interaktion auszudenken, um von einem dargestellten Ausgangszustand einen bestimmten Zielzustand zu erreichen. Für folgende Systemzustände wurden Interaktionen evaluiert:

- Anzeige des Zustandssequenzdiagramms
- Anzeige der Transitionsliste
- Anzeige der Rohdaten
- Transitionsliste verschieben
- Wechsel des Zustandssequenzdiagramms von horizontale in vertikale Ansicht
- History anzeigen
- Verbindung zwischen zwei Workbenches herstellen
- Verbindung zwischen zwei Workbenches trennen

Die Ergebnisse zu den einzelnen Systemzuständen werden im Folgenden näher beschrieben. Im Rahmen der Evaluation wurde davon ausgegangen, dass alle Visualisierungen innerhalb einer Workbench aufgerufen bzw. angezeigt werden. Diese Idee wurde in einem späteren Projektabschnitt verworfen und jede Visualisierung wird in dem aktuellen Prototypen in einer neuen Workbench dargestellt. Folglich wurden nicht alle Evaluationsergebnisse verwendet.

- **Anzeige des Zustandssequenzdiagramms:** In diesem Anwendungsfall, haben sich 40% für eine Doubletap Geste entschieden. Dabei hatten die Teilnehmer die Idee, einzelne Automaten mit einem einfachen Tap zu selektieren und mit dem doppelten Tap in das Zustandssequenzdiagramm zu wechseln. 20% der Befragten haben die Automaten mit einem Tap selektiert und wollten durch einen Button in die nächste Visualisierung wechseln. Aus den Ideen heraus wurde das Konzept umgesetzt, dass die Automaten mittels eines einfachen Tap selektiert werden können und sich daraufhin eine neue Workbench mit dem zugehörigen Zustandssequenzdiagramm öffnet.
- **Anzeige der Transitionsliste:** Um die Transitionsliste aus dem Zustandssequenzdiagramm heraus anzuzeigen, wünschten sich 20% der Probanden einen Button. Wiederum 20% der Teilnehmer würden mit einer Move Geste in die nächste Visualisierung wechseln. Außerdem würden 20% mit einem einzelnen Touch auf einen Zustand in die Transitionsliste wechseln. Viele der Probanden wollten über einen Tap auf einen Button oder auf den Zustand die Transitionsliste öffnen. Daher wurde dieser Vorschlag weiter beachtet und schließlich umgesetzt, dass die Transitionsliste durch einen Touch auf einen Zustand geöffnet wird.
- **Anzeige der Rohdaten:** Bei diesem Anwendungsfall haben sich, wie in den vorherigen beiden Fällen auch, 20% für einen Button entschieden um in die nächste Darstellung zu wechseln. 20% wählten wieder die Move-Geste um von der Transitionsliste zur Rohdatenansicht zu gelangen. Auch hier wurde der Vorgang schließlich derart gelöst, dass sich die Rohdatenansicht durch einen Tap auf eine Transition öffnet und direkt zum richtigen Zeitpunkt gesprungen wird.
- **Transitionsliste innerhalb der Workbench verschieben:** Hier hatten einige Probanden mehrere Vorschläge zur Interaktion. Zum einen würden vier Probanden die Transitionsliste mit einem Finger anwählen und durch die Drag-and-Drop Geste verschieben. Außerdem können sich fünf Probanden vorstellen die Transitionsliste mit zwei oder mehr Fingern in die gewünschte Position zu verschieben. Drei Personen würden einen oberen Rand erwarten, den sie zum Verschieben berühren können. Das Verschieben der Transitionsliste ist einer der frühen Vorschläge gewesen, die aufgrund der neuen Umsetzungsidee des Konzeptes verworfen wurden.
- **Wechsel des Zustandssequenzdiagramms von horizontale in vertikale Ansicht:** Um das Zustandssequenzdiagramm von der horizontalen in die vertikale Ansicht zu drehen, wendeten 40% der Probanden die Rotate-Geste an. Zudem erwarteten 20% der Probanden einen Button um die Ausrichtung des Zustandssequenzdiagramms zu wechseln. Aus zeitlichen Gründen wurde der Wechsel von der horizontalen in die vertikale Ansicht nicht umgesetzt.

- **History anzeigen:** Zum Anzeigen einer History erwarteten fünf Probanden einen Button. Zwei Probanden möchten durch Anwählen eines kleinen Ëxtrarands"die History aufrufen. Weitere zwei Personen würden durch die Move-Geste in vergangenen Zuständen „blättern“. Letztendlich wurde der Aufruf und das Ausblenden der History über einen Eintrag im Menü realisiert.
- **Verbindung zwischen zwei Workbenches herstellen:** Bei diesem Anwendungsfall haben einige Probanden mehrere Ideen zur Umsetzung gehabt. Sieben Probanden haben sich überlegt, die zu verbindende Workbench mit einem Finger zu selektieren und daraufhin durch die Move-Geste zur zweiten Workbench eine Interaktion herzustellen. Zwei Probanden erwarteten eine Linie zwischen den Workbenches zu ziehen und sobald der Rand der zweiten Workbench berührt wird, sollte sich ein Menü öffnen, in dem die Art der Verbindung ausgewählt werden kann. Außerdem erwarteten zwei Probanden einen Button zur Erstellung einer Verbindung. Das Konzept der Synchronisation wurde derart gelöst, dass mittels eines Menüs das Synchronisationsmenü aufgerufen werden kann.
- **Verbindung zwischen zwei Workbenches trennen:** Um die Verbindung zwischen zwei Workbenches zu trennen, würden zwei Probanden die Visualisierung aus der Workbench in einen freien Bereich des Tisches ziehen. Weitere zwei Probanden würden auf die Verbindungsanzeige einer Workbench einen Tap ausführen um die Verbindungsoptionen zu öffnen und die Konnektivität zu trennen. Zudem würden zwei Personen, falls eine sichtbare Verbindungslinie existiert, diese einfach entgegengesetzt ihres Verlaufs durchstreichen.

12.5 Abschlussevaluation

Während des gesamten Projektes wurden zwei Evaluationen sowie drei Vorstellungen mit externen Personen durchgeführt. Die zwei Evaluationen hatten zum einen das Ziel das Visualisierungs- sowie das Interaktionsdesign (siehe Kapitel 12.2.2) zu evaluieren und zum anderen den gesamten Prototypen zu evaluieren, um die Gebrauchstauglichkeit abermals zu erhöhen. Die drei Präsentationen fanden zum einen bei der langen Nacht der Wissenschaft (siehe Kapitel 12.4.4), dem Schülerinformationstag und abschließend bei BMW in München statt (siehe Kapitel 12.5.6). Diese hatten bspw. den Zweck verschiedene Eindrücke von externen Personen zu erlangen. In diesem Kapitel wird die Abschlussevaluation genauer beschrieben.

Wie im Kapitel 5.5 bereits näher beschrieben, besteht der letzte Schritt jeder Iteration im SCiVA-Prozesses aus einer Evaluation. Diese Evaluation prüft das gesamte System und geht der Aufgabe nach, die Gebrauchstauglichkeit des Systems zu erhöhen. Dazu sind die Probanden unter anderem während der Evaluation dazu aufgefordert worden,

ihre Gedankengänge laut auszusprechen. Diese wurden protokolliert, um so ein besseres Verständnis der aktuellen Tätigkeit zu erlangen und ein genaueres Evaluationsergebnis zu erzielen. Zusätzlich wurden die Interaktionen auf dem Tisch per Videokamera aufgezeichnet und zur genaueren Auswertung genutzt. Der Zeitraum der Evaluation war von 23.02.2011 bis zum 25.02.2011. Ferner wurden die 20 Probanden in Zweierteams eingeteilt, denn es wurde außerdem das kollaborative Arbeiten mit Hilfe des Prototyen evaluiert.

12.5.1 Aufbau

Der Aufbau wird in diesem Kapitel in die drei Bereiche Hardware, Software und Dokumente aufgeteilt und näher erläutert.

Hardware: Die für die Evaluation genutzte Hardware setzt sich aus nahezu den gleichen Gegenständen zusammen, wie bei der ersten Evaluation. Die Abbildung (Abb.: 12.10) verdeutlich den allgemeinen Aufbau. Zum Einen bestand der Aufbau aus einer Kamera(1), welche auf einem Stativ befestigt wurde, um die gesamten Interaktionen der Probanden aufzuzeichnen. Diese Art der Protokollierung diente zur späteren, detaillierteren Auswertung der Ergebnisse. Zum Anderen wurde der Multitouch Tisch(2) genutzt, auf dem der Prototyp ausgeführt wurde und die Teilnehmer interagiert haben. Zusätzlich wurde ScreenCapture auf dem Multitouch Tisch ausgeführt, dadurch konnten die ausgeführten Gesten besser betrachtet werden.

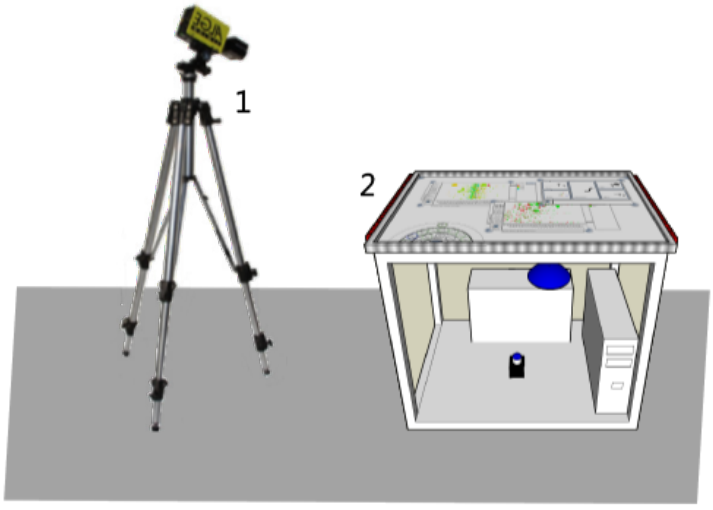


Abbildung 12.10: Aufbau der genutzten Hardware

Software: Für die Evaluation wurde der bis dahin aktuellste Stand des Prototypen genutzt, was einem sehr weit fortgeschrittenen Status entsprach. In Folge dessen konnten die meisten Funktionen während der Evaluation abgefragt werden.

Dokumente: Verständlicherweise sind nicht nur die Hardware sowie die Software wichtige Aspekte für die erfolgreiche Durchführung der Evaluation gewesen. Gleichermaßen stellten die angefertigten Dokumente eine wichtige Eigenschaft dar, damit die erfolgreiche Durchführung gewährleistet werden konnte. In diesen Zusammenhang wurden folgende Dokumente erstellt:

- Zu allererst ein Dokument, welches grob die Ziele sowie weitere Eckdaten der Evaluation beschreibt.
- Weiterhin ein Dokument, welches den schrittweisen Ablauf der Evaluation wiedergibt.
- Eine Doodle-Umfrage zur genauen Einplanung der Zeitslots und zusätzlich ein entsprechendes Dokument.
- Darüber hinaus ein Dokument zur Einladung der Probanden und eine Datenschutzerklärung.
- Des Weiteren verschiedene Dokumente zur Befragung von statistischen Daten wie bspw. das Alter oder das Geschlecht (demographische Daten, SUS-Fragebogen).
- Überdies die Beschreibung der fünf durchzuführenden Evaluationsaufgaben.
- Außerdem ein Dokument welches dem Evaluationsteam einige Hinweise zur Durchführung gibt.
- Darauf folgend einige Dokumente zur schnelleren Auswertung der statistischen Daten sowie Vorlagen für die Protokolle.
- Abschließend mehrere Dokumente die zur Auswertung der Protokolle und Videos dienten.

12.5.2 Informationen zu den Probanden

An der Evaluation haben insgesamt 20 Probanden teilgenommen. Diese setzten sich aus 80% männlichen und 20% weiblichen Probanden zusammen. Für eine Evaluation im technischen Bereich ist ein Schnitt von 20% weiblichen Probanden ein guter Wert. Die Einladung galt dabei nicht gezielt einer bestimmten Personengruppe, sondern wurde an verschiedene potenzielle Personen geschickt. So wurde unter anderem der Verteiler der Fachschaft Informatik an der Universität genutzt, wie auch ein Eintrag in verschiedenen Vorlesungsforen getätigt. Weiterhin wurden weitere Studierende angeschrieben und auch im persönlichen Umfeld der Projektgruppe nach potenziellen Probanden gesucht.

Aber auch von den Mitarbeiterinnen und Mitarbeitern im OFFIS zeigten sich einige an der Evaluation interessiert.

Dadurch konnten verschiedene Altersgruppe erreicht werden, dies ist weiterhin in der Abbildung (Abb.: 12.11) dargestellt. Der Altersdurchschnitt von allen Probanden lag bei 25,5 Jahren, wobei die Alter 22, 24 und 32 mit insgesamt 55% den Großteil darstellen.

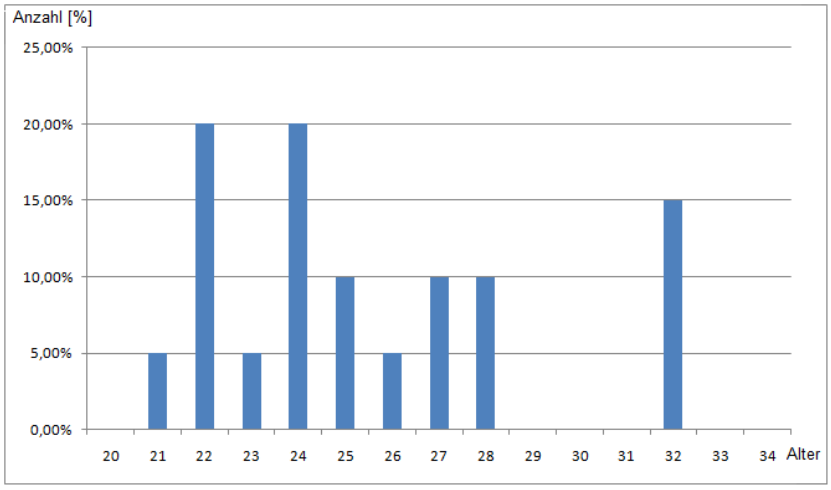


Abbildung 12.11: Alter der Probanden

Der Abbildung 12.12 ist interessanterweise zu entnehmen, dass 40% der Probanden deren eigene Multitoucherfahrung auf einer Skala von null (keine Erfahrung) bis fünf (sehr viel Erfahrung) mit dem Wert drei bewerteten. Im Gesamtdurchschnitt wurde die Multitoucherfahrung aller Probanden jedoch mit dem Wert 2,3 angegeben.

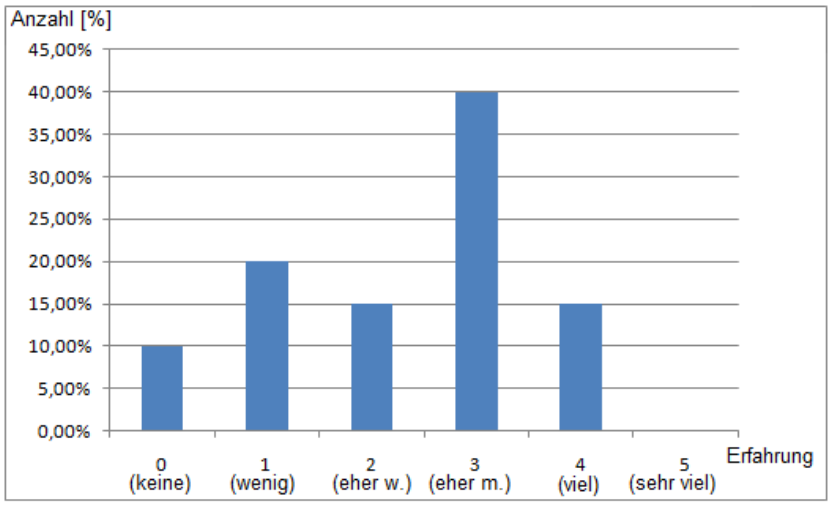


Abbildung 12.12: Multitoucherfahrung der Probanden

12.5.3 Ablauf der Evaluation

Einleitend zur Evaluation sind die potenziellen Probanden eingeladen worden. Dazu wurde ein Doodle-Link den jeweiligen Einladungen angefügt, in dem die Probanden mögliche Termine auswählen konnten. Eine Mehrfachauswahl von Terminen wurde ermöglicht, damit jeder Proband an der Evaluation teilnehmen konnte. Da die Dauer einer einzelnen Evaluationsstudie mit einer Stunde einkalkuliert wurde, erhielten die Probanden eine Aufwandsentschädigung für die erfolgreiche Teilnahme. Im Durchschnitt nahm eine Evaluationsstudie dabei 50 Minuten in Anspruch. Nachdem die potenziellen Probanden in die Doodle-Umfrage eingetragen wurden oder sich selbst eingetragen hatten, haben wir die einzelnen Evaluationspaare zusammengestellt und die Probanden über den jeweiligen Zeitslot in Kenntnis gesetzt.

Der weitere Ablauf der Evaluation war wie folgt: Zu allererst sind die Probanden gebeten worden das Dokument über die Datenschutzerklärung sowie die demographischen Daten (Alter, Geschlecht, Multitoucherfahrung, usw.) auszufüllen. Anschließend konnten die beiden Probanden einen Platz um den Multitouch Tisch wählen, von dem aus sie arbeiten wollten und die Videoaufnahme sowie das ScreenCapture wurde gestartet. Darauffolgend wurden allgemeine Hinweise über die Projektgruppe und dem Tisch gegeben. So konnten die Probanden die Hintergründe des zu evaluierenden Prototypen besser verstehen. Daran anschließend ist eine Selbstexperimentierphase ermöglicht worden, in der die Probanden ohne jegliche Hilfestellungen oder Aufgaben den Prototypen bedienen konnten. Unterdessen wurden die interessanten Aspekte protokolliert, zum Beispiel in welche Richtung die Probanden die Workbench öffneten, ob das Menü leicht zu finden ist oder ob die Probanden sich gegenseitig geholfen bzw. untereinander kommuniziert haben. Nachfolgend ist den Probanden die Anwendung näher erklärt worden, damit diese den gesamten Ablauf verstehen. In diesem Kontext wurden die Zusammenhänge zwischen den verschiedenen Visualisierungen erklärt, die Funktionen des Programms vorgestellt sowie der Aufruf des Menüs gezeigt. Weiterhin wurde der Zweck, für den der Prototyp dient, verdeutlicht. Als die Probanden diese Zusammenhänge gut verstanden hatten, wurden die Aufgaben bearbeitet. Dazu wurde immer ein Aufgabenzettel verteilt, den die Probanden durchführen sollten. Diese fünf Aufgabenzettel behandelten folgende Thematiken:

- Die Interaktion in den verschiedenen Visualisierungen.
- Die Synchronisation verschiedener Workbenches.
- Die Bedienung der History.
- Die Interaktion mit der Filter Funktion.
- Sowie eine Aufgabe, die das kollaborative Arbeiten in den Vordergrund stellte.

Bei der Ergebnisprotokollierung wurde beispielsweise darauf geachtet, welche Gesten die Probanden anwenden, wie schnell einzelne Aufgaben bearbeitet wurden bzw. Funktionen bedient werden konnten, wie das allgemeine Verhalten der Probanden untereinander ist und welche weiteren Ideen sowie Verbesserungen die Probanden zu dem Prototyen nannten. Nach den bearbeiteten Aufgabenstellungen wurden die Probanden gebeten die System Usability Scale (SUS) auszufüllen und konnten anschließend eine der angebotenen Aufwandsentschädigungen wählen.

12.5.4 Ergebnisse

Innerhalb dieses Abschnitts werden einige Ergebnisse der Evaluation erläutert.

Workbench erstellen: Beim Erstellen einer Workbench hatten die Probanden verschiedene Möglichkeiten ausprobiert. In diesem Rahmen wurden verschiedene Gesten angewendet, beispielsweise ein Kreis, DoubleTap, Tap sowie die Pinch Geste, um nur einige zu nennen. Am häufigsten wurde allerdings ein Strich in verschiedene Richtungen gezogen, wobei der Strich zum eigenen Körper am seltensten ausgeführt wurde. Mehrfach wurde ein Strich von der Tischkante nach innen gezogen, was ein interessantes Ergebnis darstellte. Denn dies war zuvor andersrum implementiert worden. Die korrekte Strichrichtung wurde jedoch spätestens bis zum Ende der Evaluation gelernt.

Workbench schließen: Das Schließen der Workbench wurde zu Beginn der Evaluation oft nicht selbstständig gefunden, da die Probanden nicht erkannten, wie das Menü geöffnet wird oder gar nicht wussten, dass ein Menü existierte. Weiterhin sind die Probanden oft der Meinung gewesen, dass Schließen über das Menü zu lange dauert und stattdessen eine andere Möglichkeit zum Schließen gegeben sein sollte. Einer der genannten Möglichkeiten war zum Beispiel ein „X“ in der oberen Ecke, wie die Windowsnutzer es gewohnt sind. Allerdings gehörte das Schließen der Workbench über ein Menü zum Designprozess. Des Weiteren beinhalten die anderen Möglichkeiten weitere Problematiken. So könnte ein „X“ auf dem Rand dazu führen, dass ein Analyst zufällig beim Rotieren einer Workbench dieses berührt und dadurch schließt. Andere Ideen, damit die Workbenches schneller geschlossen werden können, sind bspw. folgende gewesen: Die Workbench aus den sichtbaren Bereich rausschieben, DoubleTap auf den Rand und die Workbench so lange verkleinern, bis diese verschwindet.

Workbenchhandling: Am Anfang der Evaluation war nicht allen Probanden direkt ersichtlich, dass das Workbenchhandling (Rotieren, Verkleinern, Vergrößern, Verschieben) auf dem Rand ausgeführt werden muss. Spätestens bis zum Ende der Evaluation wurde dies jedoch von allen Probanden erkannt. Meistens wurden zwei gegenüberliegende Kanten oder Ecken zur Manipulation genutzt. Dadurch ergaben sich allerdings Probleme, wenn Teile des Randes nicht mehr sichtbar waren, weil diese bspw. außerhalb des

sichtbaren Bereiches lagen oder von einer anderen Workbench überlagert wurden. Vereinzelt ist die Workbench auch anders manipuliert worden. So wurde dies manchmal mit nur einer Hand durchgeführt oder mit einen Finger auf dem Rand und einen anderen Finger bspw. in der View oder außerhalb der Workbench.

Menü: Während der Evaluation fiel auf, dass das Menü selbständig schwer zu finden ist. Dies lag unter anderem auch daran, dass die Zeit bis das Menü aufpoppte ein wenig zu lang eingestellt war. Weiterhin wurde dieser Punkt auch von einigen Probanden bemängelt, da diese zwar den Finger längere Zeit auf dem Rand gehalten hatten, jedoch nicht so lange, bis sich das Menü öffnete. Dadurch wurde vermutet, dass das Menü anders geöffnet werden muss. Anfängliche Fehler bei der Bedienung des Menü waren unter anderem, dass der Finger innerhalb des Menüs nicht gezogen wurde, sondern losgelassen wurde und versucht wurde daraufhin einen Eintrag zu betätigen oder mit zwei Fingern im Menü zu interagieren. Bis zum Ende der Evaluation konnten allerdings alle das Menü gut bedienen, Probleme gab es lediglich noch durch Fehleingaben. Dass das Menü außerhalb des sichtbaren Bereiches gelangt und sich immer weiter nach unten bewegt, sobald die Zurück-Funktion betätigt wurde, ist von den Probanden als störend empfunden worden. Zudem fiel störend auf, dass die Hand öfters die Menüeinträge überdeckte und dadurch diese nicht immer lesbar waren. Überdies wurde der Vorschlag gegeben, dass beim Berühren des Randes ein Feedback (z. B. ein sich aufbauender Kreis) darüber gegeben wird, dass in kurzer Zeit das Menü geöffnet wird. Aber auch, dass ein Überblick über die Hierarchieebenen im Menü gegeben werden sollte, wurde als Vorschlag genannt.

History: Nachdem die die Bedienung des Menüs erkannt wurde, ist auch die Funktion der History gefunden worden. Zudem war die Funktion leicht zu verstehen, wobei die History selten angewendet worden ist. Wie auch bei den anderen Ergebnissen, hatten die Probanden auch bei der History weitere Vorschläge oder andere Erwartungen an der Bedienung. So ist unter anderem erwartet worden, dass verschiedene View in die geöffnet History geschoben werden konnten, und dadurch zu einem späteren Zeitpunkt wieder aufgerufen zu werden. Weiterhin sind die Probanden häufiger zuerst davon ausgegangen, dass die History für den gesamten Analysepfad gilt und nicht nur für eine Workbench. Zum Schließen der History ist des Weiteren der Vorschlag von den Probanden geäußert geworden, dieses durch ein Reinschieben in die Workbench zu realisieren oder durch eine Press and Hold Geste.

Synchronisation: Als ein Ergebnis der Synchronisation wurde bemängelt, dass zuerst eine ungenutzte Workbench erstellt werden muss, bevor die Synchronisation durchgeführt werden kann. Dabei ist ein Vorschlag gewesen, dass direkt eine Workbench mit den richtigen Visualisierungen erstellt wird, sobald auf „Copy“ (nach der Evaluation in „Kopieren“ geändert) ein Touch ausgeführt wird. Weiterhin ist von den Probanden angemerkt worden, dass die Funktion aufgrund der Tischgröße nicht benötigt wird, da es einfacher wäre dem anderen Probanden die eigene Workbench zu zeigen. Daher ist von

den Probanden die Anmerkung gegeben worden, dass die Funktion der Synchronisation mehr Sinn auf verschiedenen, räumlich getrennten Tischen macht. Zudem führten die Bezeichnungen der Buttons (Copy, Mirror) zu Fehlinterpretationen und damit zu Anwendungsproblemen.

Filter: Die Filterfunktion war offensichtlich schwer zu verstehen, da mehrere Probanden nicht wussten, wie und wo die Filter angewendet werden. Des Weiteren ist von einigen Probanden erwartet worden, dass irgendwo (bspw. im Rand) eine Markierung besteht, dass ein Filter gesetzt ist. Dies soll die Übersichtlichkeit erhöhen. Weitere Vorschläge waren, dass Filter selbst benannt werden sollten, eine Filteransicht existiert, die gleichzeitig mit der Automatenansicht geöffnet ist und in der direkt Filter de-/ selektiert werden können. Ebenfalls ist von den Probanden versucht worden, die Automatenansicht zu filtern, indem ein Filter dort hineingezogen wird.

Visualisierungen: Bei den Visulisierungen fiel auf, dass an verschiedenen Stellen ein Feedback fehlte, bspw. in welcher Visualisierung der Proband sich aktuell befindet und was gerade passiert (z. B. wenn ein Ladevorgang länger dauert). Darüber hinaus ist anhand der Verbindungslinien nicht ersichtlich, welches Zustandssequenzdiagramm zu welchen Automaten gehört, das gleiche gilt auch für die weiteren Visualisierungen. Ebenso haben Probanden vereinzelt versucht die Visualisierungen über das Menü zu wechseln. Da eine Analyse mit dem Prototypen allerdings einer bestimmten Abfolge folgt, ist dieser Wechsel nicht vorgesehen. Im Zustandssequenzdiagramm wurde nach einer Möglichkeit gesucht, den Zeitpunkt der Zustandsübergänge genauer ablesen zu können. Dabei ist unter anderem erwartet worden, dass durch den Tap auf die Zeitachse oder durch eine gezogene Linie vom Startpunkt eines Zustandes zur Zeitachse. Diese Linie könnte weiterhin verschoben werden, damit auch andere Zeitpunkte genauer abgelesen werden können. Einmal ist auch vorgeschlagen worden, dass die Bezeichnung der Zustände direkt in den Zuständen stehen sollte und nicht nur in der Spaltenüberschrift. Das Sortieren und Neuanordnen der Spalten in der Rohdatenansicht ist hingegen als positiv empfunden worden. Dennoch hatten viele Probanden Probleme zu verstehen, warum die Zeitpunkte in der Rohdatenansicht nicht immer mit denen der anderen Visualisierungen übereinstimmen.

Kollaboration: Da der Multitouch-Tisch von allen Seiten bedienbar ist, ist es auch interessant gewesen, wie die beiden Probanden sich um den Tisch aufstellen. Diesbezüglich ergab sich das Ergebnis, dass die meisten Probanden während der gesamten Studie ihre erste Position behalten haben. Standen die Probanden dabei nebeneinander, wurden die Lösungen zu den Aufgaben meist zusammen erarbeitet. Weiterhin wurden neue Erkenntnisse verbal ausgetauscht, aber auch Lösungswege von dem anderen Probanden abgeschaut. Sofern die Aufgabenstellung ergab, dass etwas verglichen werden soll, sind die verschiedenen Workbenches häufig nebeneinander gelegt worden.

Allgemeines: Der Großteil der Funktionalität ist im Laufe der Evaluation erlernt worden und konnte am Ende der Evaluation gut angewendet werden. In diesem Zusammenhang passierten immer weniger Fehleingaben, je weiter die Evaluation fortgeschritten war. Zwei Vorschläge die außerdem von den Probanden genannt wurden waren, dass es eine Möglichkeit geben sollte alle verbundenen Workbenches gleichzeitig zu verschieben und dass die Verbindungslinien zwischen den Workbenches einen Pfeil besitzen sollten, um die Richtung besser darzustellen.

12.5.5 Verbesserungen aufgrund der Evaluation

Die im Abschnitt 12.5.4 dargestellten Evaluationsergebnisse stellen einen Teil aller Ergebnisse dar. Im Folgenden werden einige Änderungen aufgelistet, die nach dem Abschluss der Evaluation, aufgrund der Ergebnisse, durchgeführt wurden.

- Die Bezeichnung „all“ zum Zurücksetzen aller Filter, ist in „<ZURÜCKSETZEN>“ geändert worden. Diese Bezeichnung beschreibt die eigentliche Funktion besser.
- Sobald ein Filter gesetzt wird erscheint nun ein Hinweis, dass ein Filter gesetzt wurde und wo dieser Filter angewendet werden kann.
- Die Bezeichnung „...“, um die jeweilige Aktion beim Filter setzen und in der XML-Auswahl abzubrechen, wurde durch die Bezeichnung „<ABBRUCH>“ getauscht.
- Die Strichrichtung, um eine Workbench zu öffnen ist um 180° gedreht worden. Zuvor musste ein Strich von der Tischmitte zum eigenen Körper gezogen werden, damit die Workbench richtig ausgerichtet wird. Der neue Vorgang sieht vor, dass der Strich von der Tischkante zur Tischmitte gezogen werden soll. Dieser Vorgang ist angenehmer für die Anwender.
- Sofern der Anwender die ganze Anwendung beenden will (Interaktion im Menü: Programm –> Beenden), erscheint eine Abfrage, ob dies auch wirklich gewollt ist. Vor dieser Änderung wurde direkt die ganze Anwendung beendet, sobald im Menü der Eintrag „Beenden“ betätigt wurde. Dies führte zu einigen Missverständnissen. Beispielsweise wollten einige Probanden lediglich eine Workbench schließen und kamen auf den Eintrag „Beenden“, wodurch direkt die ganze Anwendung geschlossen wurde.
- Im Zustandssequenzdiagramm sind die Einheiten der Zeitskala (min/sec/ms) als Spaltenüberschrift hinzugefügt worden. Dieser Wunsch wurde von einigen Probanden geäußert.

- Vor und während der Evaluation hatten einige Zustandsüberschriften eine schwarze Schriftfarbe auf einen blauen Hintergrund. Während der Evaluation wurde mehrfach angemerkt, dass dies nicht immer gut lesbar ist und ist daher verbessert worden.
- Weiterhin ist der Aufruf des Menüs dahingehend verbessert worden, dass sich das Menü nun schneller öffnet. Dazu wurde die Zeit auf eine halbe Sekunde verringert.
- Beim Menü sind zudem weitere Verbesserungen durchgeführt worden, um die Bedienung zu verbessern.
- Eine weitere sinnvolle Verbesserung ist, dass alle wichtigen Bezeichnungen und Beschreibungen in die deutsche Sprache übersetzt worden sind. So sind die Synchronisationsfunktionen „Copy“ in „Kopieren“ und „Mirror“ in „Verknüpfen“ geändert worden, sowie die Checkbox „with Childs“ in „mit Pfad“. Aber auch die anderen Übersetzungen, z. B. im Menü, sollen den Anwendern die Bedienung des Systems erleichtern.
- Die Funktion „Bookmark“ ist dabei in die Bezeichnungen „Filter anwenden“ und „Filter setzen“ umbenannt worden.
- Außerdem existiert beim Export in der Rohdatenansicht nun ein Feedback darüber, ob und wohin der Export gespeichert wurde.
- Bei der Synchronisationsdarstellung ist weiterhin eine Beschreibung hinzugefügt worden, die den Anwender darauf hinweist, dass zuerst eine ungenutzte Workbench erstellt werden muss.
- Zudem sind viele Funktionen bzw. Buttons durch verschiedene Bilder visualisiert worden, sodass der Anwender bereits optisch auf die jeweilige Funktion hingewiesen wird und diese leichter versteht.

12.5.6 Weitere Präsentationen

Bevor der erstellte Prototyp bei BMW in München präsentiert wurde, ist dieser zuvor, am 28.02.2011, auf dem Schülerinformationstag Informatik [Old11] der Universität Oldenburg vorgestellt worden. Das Ziel der Präsentation auf dem Schülerinformationstag war unter anderem, den Schülern und weiteren Besuchern den erstellten Prototypen vorzustellen und zu sehen, wie diese die Bedienung des Prototypen empfinden. Dazu wurde die Anwendung auf einem HP TouchSmart, einem Multitouch-PC, präsentiert. Die Besucher des Standes konnten so mit dem System interagieren.

Im März 2011 fand zudem eine Fahrt nach München zu BMW statt. Die dort präsentierte Anwendung enthielt bereits die meisten Verbesserungen und Änderungen,

die aufgrund der Evaluation durchgeführt wurden. Daher konnte ein sehr guter Stand der Anwendung präsentiert werden. Diese Anwendung wurde, wie auch zuvor auf dem Schülerinformationstag, auf dem HP TouchSmart präsentiert und konnte von den BMW-Mitarbeitern getestet bzw. bedient werden. Nachdem die Anwendung selbstständig bedient werden konnte, äußerten die Mitarbeiter große Zufriedenheit mit dem Prototypen. Anschließend zur Anwendungspräsentation sind innerhalb eines Fokusgruppengesprächs einge Fragen durchgegangen worden. Diese Fragen behandelten unter anderem die Anforderungen von BMW und einen möglichen Produktiveinsatz.

Kapitel 13

Projektumsetzung

In den folgenden Kapiteln wird erläutert, wie wir die erhobenen Anforderungen an Funktionalität, Design und Bedienbarkeit umgesetzt haben. Dazu werden die grundlegenden verwendeten Technologien erläutert und eine Einführung in das Model-View-ViewModel Architekturmodell gegeben. Dieses Modell stellt die zentrale Architektur der Anwendung dar und wurde an die Anforderungen angepasst.
Im Kapitel Module werden die wesentlichen Systemkomponenten hinsichtlich ihrer Implementierung beschrieben. Dabei wird auch auf die Einbindung der Module im Programmkontext sowie deren Schnittstellen eingegangen.
Abschließend erfolgt eine Beschreibung der Umsetzung des KnoVA-Modells, welches ermöglicht das Wissen der Analysten zu extrahieren und anderen Analysten zur Verfügung zu stellen. Dafür wird auch auf die standardisierte Parametrisierung der Visualisierungen eingegangen.

13.1 Verwendete Technologien

Dieser Abschnitt beschreibt, welche Technologien im Einzelnen bei der Projektumsetzung verwendet wurden.

WPF

Wie bereits in den Grundlagenkapiteln beschrieben und im weiteren Verlauf des Berichts erwähnt, wurde primär die Windows Presentation Foundation verwendet. Die Wahl fiel auf WPF, da es recht viele Möglichkeiten bietet, eine Anwendung mit den genannten Anforderungen zu implementieren und bereits von Haus aus viele Hilfsmittel zur Verfügung stellt. Dazu zählen insbesondere die Unterstützung von MultiTouch, das Bereitstellen verschiedener *Events* zur Registrierung und Verarbeitung von Touch-Ereignissen verschiedenster Art und die einfache Möglichkeiten, eine effiziente GUI zu

erstellen - sowohl durch vorgefertigte, als auch selbst definierte Elemente.

Durch diese Hilfen war es möglich, den Fokus unserer Arbeit auf die funktionale Umsetzung der Anforderungen, sowie das Design der Anwendung zu legen. An den meisten Stellen konnten die von WPF zur Verfügung gestellten Mechanismen für die Anwendungssteuerung verwendet werden und es musste nur vergleichsweise wenig Arbeit dafür aufgebracht werden, die Schnittstelle zwischen Anwender und Programm zu modifizieren, so dass sie den Ansprüchen genügt.

Visual Studio 2010

Die gewählte Entwicklungsumgebung war *Visual Studio 2010* von Microsoft. Die Wahl fiel recht schnell darauf, da durch die Verwendung von WPF als Programmiersprache C# empfehlenswert schien und Visual Studio 2010 sowohl C#, als auch WPF unterstützt.

Neben den Grundfunktionalitäten von Visual Studio 2010 wurden zudem die AddOns *ReSharper* und *AnkhSVN* verwendet.

ReSharper Der ReSharper (R#) von *JetBrains* wurde in erster Linie als Codeanalysewerkzeug verwendet, um dadurch die Qualität des Quellcodes zu sichern und den Code einheitlich zu halten. Durch die einfache Konfigurierbarkeit des R# konnten, neben den für C# üblichen Codestandards, auch einige eigene definiert werden. Die Verwendung von R# war notwendig, da Visual Studio 2010, im Gegensatz zu vielen anderen Entwicklungsumgebungen, nur sehr bedingt Refactoring und automatische Codekorrektur unterstützt. Der R# wurde mit der *Academic License* verwendet.

AnkhSVN Mit AnkhSVN von *CollabNet* war es möglich, Visual Studio 2010 mit einem SVN Repository zu verbinden. Leider bietet AnkhSVN nur die reine Verbindung zu einem SVN an. Für das Auflösen von Konflikten muss ein externes Programm genannt werden. In Falle dieses Projekts war das der Konflikteditor von TortoiseSVN.

Multi-Touch Vista

Bei *Multi-Touch Vista* von *CodePlex* handelt es sich um einen Simulator für Touch-Eingaben. Dabei werden Ereignisse, die von der Maus ausgelöst werden, so an WPF Windows gesendet, dass sie als Berührungen interpretiert werden. Multi-Touch Vista wurde vorwiegend für Windows Vista entwickelt, unterstützt in der neusten Version

jedoch auch Windows 7. Zudem unterstützt es die Verwendung von mehreren Mäusen.

Die Verwendung von Multi-Touch Vista war ratsam, da so die Anwendung auf jedem Rechner getestet werden konnte und kein Gerät mit Berührungserkennung notwendig war. Außerdem konnte durch die Verwendung mehrerer Mäuse die Unterstützung von Multi-Touch getestet werden. Die Simulation war notwendig, weil WPF bestimmte Ereignisse nur für Berührungen zur Verfügung stellt, die mit einer Maus nicht getestet werden konnten. Außerdem wurde bei der Entwicklung festgestellt, dass es sich auch umgekehrt verhält, also Dinge, die mit Maus funktionierten nicht notwendigerweise auch mit Berührungen kompatibel waren.

13.2 Model-View-ViewModel Architektur

Das von uns entwickelte System nutzt als Architekturgrundlage das *Model-View-ViewModel*(MVVM) Entwurfsmuster. In diesem Unterkapitel soll das Konzept von MVVM sowie unsere Umsetzung und Anwendung des Musters beschrieben werden.

Das Model-View-ViewModel Muster ist ein Entwurfs- oder besser Architekturmuster, das bei der Entwicklung von *WPF* oder *Silverlight* Anwendungen genutzt werden kann. Die grundlegende Idee hinter MVVM ist die Trennung von Daten, Anwendungslogik und Bedienoberfläche. Damit ähnelt es allgemein bekannten Architekturmustern wie *Model-View-Controller* (MVC) und *Model-View-Presenter* (MVP), wobei eine von Martin Fowler veröffentlichte Weiterentwicklung des MVP-Musters, das *Presentation Model* (PM) Muster, die Grundlage von MVVM bildet. MVVM kann als eine Spezialisierung dieses Musters betrachtet werden, die an WPF und dessen Stärken angepasst ist. [Ell10, S. 1249f.]

13.2.1 Konzept

MVVM unterscheidet im Kern zwischen *Model*, *View* und *ViewModel*. Das Model realisiert, analog zum MVC-Muster, die Datenrepräsentation der Applikation. Die View ist, auch wiederum analog zum MVC-Muster, die Benutzeroberfläche des Programms. Sie kümmert sich um die Darstellung von Daten aus dem ViewModel und die Weiterleitung von Benutzerinteraktionen. Das ViewModel kann als eine Art Abstraktion einer Ansicht betrachtet werden. Es stellt die anzuzeigenden Daten und Befehle zur Ausführung von Anwendungslogik zur Verfügung, die von der entsprechenden View genutzt werden können. [Smi09][Ell10, S. 1250.] Abbildung 13.1 gibt einen Überblick der Struktur von MVVM. Sie beinhaltet die Eigenschaften der einzelnen Elemente sowie deren Beziehungen untereinander, worauf im Folgenden näher eingegangen wird.

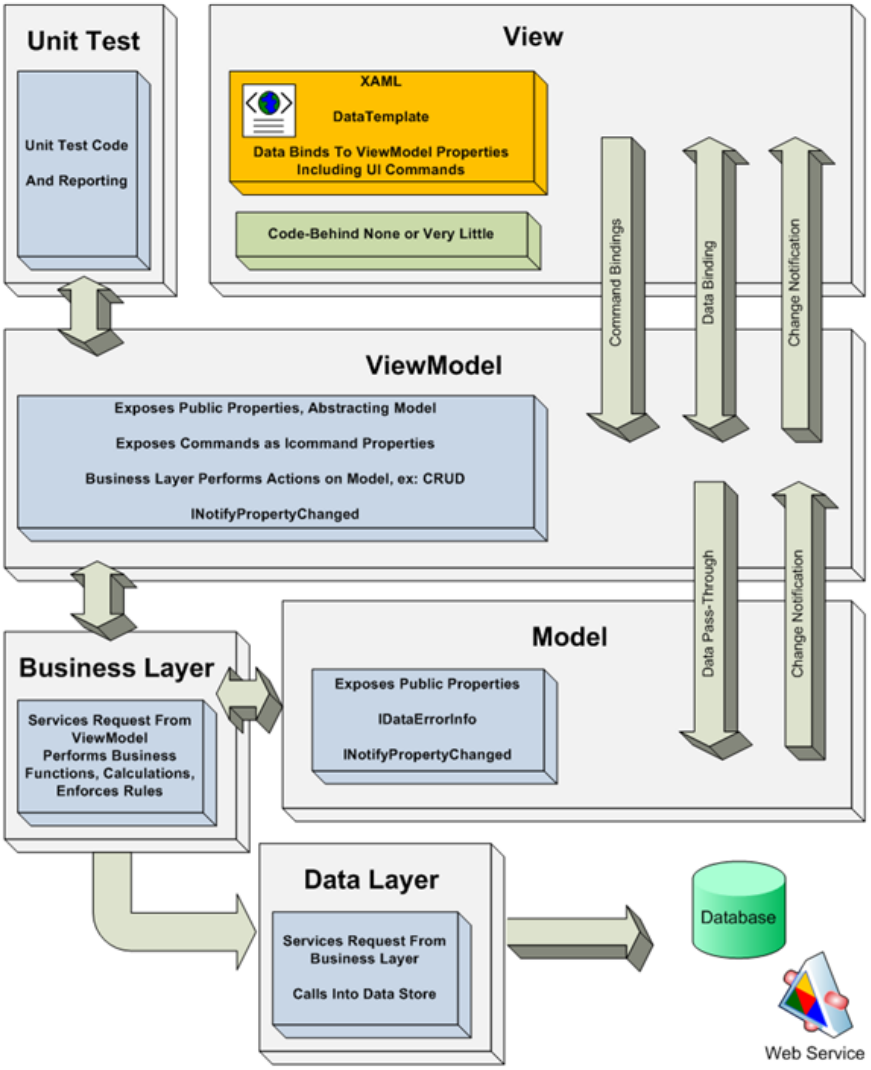


Abbildung 13.1: Erweiterte Model-View-ViewModel Architektur [Shi08]

Data Binding

Ein zentrales Feature, das MVVM so nützlich macht, ist das *Data Binding*, welches von WPF zur Verfügung gestellt wird. Hierbei wird eine View an die *Properties* eines ViewModels gebunden. Ändert sich der Wert eines Properties, so wird über das Data Binding direkt die entsprechende View aktualisiert. Umgekehrt können Benutzereingaben, die in der View gemacht werden, direkt in das ViewModel übertragen werden, wobei auch eine Validierung der Eingaben möglich ist. Über das Data Binding werden View und ViewModel synchron zueinander gehalten, ohne dass im ViewModel expliziter Code zur Aktualisierung der View geschrieben werden muss. [Smi09]

Commands

Ein weiteres wichtiges Feature, das zur Realisierung von MVVM verwendet wird, sind *Commands*. Commands ermöglichen einer View auf die Funktionalität eines ViewModels zuzugreifen. Das ViewModel stellt dafür die entsprechenden Commands als Properties zur Verfügung, die dann über das Data Binding von der View genutzt werden. [Smi09]

Lose Kopplung

Charakteristisch für MVVM ist die starke Trennung von View, Model und ViewModel.

Das Model ist dabei vollständig unabhängig von den anderen Komponenten und kennt diese nicht. Zustandsänderungen werden über *Change Notification Mechanismen* an das ViewModel propagiert. Das ViewModel kennt das Model und kann damit auf dessen Daten zugreifen und diese ggf. manipulieren.

Das eigentlich Interessante am MVVM Muster ist jedoch die starke Trennung zwischen View und ViewModel. Das ViewModel als Abstraktion einer Ansicht, stellt über Properties und Commands Daten und Funktionen zur Verfügung. Diese können von konkreten Views genutzt werden, ohne dass die Views dem ViewModel bekannt sind. Die Synchronisation geschieht dabei automatisch über das bereits erwähnte Data Binding. Wie in WPF Applikationen üblich werden die Views in XAML, einer deklarativen Sprache zur Erstellung von Bedienoberflächen, definiert. Das besondere im Kontext von MVVM ist, dass die Views möglichst komplett in XAML, das heißt ohne *Code-Behind* geschrieben werden. Die Verknüpfung einer View mit einem ViewModel erfolgt über den sogenannten *DataContext*, ein spezielles Property der View. Der DataContext kann entweder explizit gesetzt oder über *Data Templates* zur Laufzeit bestimmt werden. Data Templates beschreiben, welche konkrete View auf ein

ViewModel angewendet wird. Die Templates werden ebenfalls in XAML definiert und zur Laufzeit vom *Ressourcensystem* angewandt. [Smi09]

Durch die starke Trennung von View und ViewModel ist eine separate Entwicklung dieser Komponenten möglich. Dazu müssen lediglich die vom ViewModel zur Verfügung gestellten Properties und Commands bekannt sein. Neben der getrennten Entwicklung erlaubt dies auch den einfachen Austausch von Views, ohne das andere Programmteile davon betroffen sind. [Smi09]

Erweiterte Architektur

Die in Abbildung 13.1 gezeigte erweiterte MVVM Architektur enthält neben Model, View und ViewModel noch einen *Business Layer* und einen *Data Layer*. Der Business Layer enthält Funktionen die über den Funktionsumfang der ViewModels, als abstrahierte Ansichten, und des Models, als Datenrepräsentation, hinausgehen. Der Business Layer kann von ViewModel und Model genutzt werden und ist für den Austausch von Daten vom Data Layer und Model zuständig. Im Data Layer sind die Funktionen zum Zugriff auf die benötigten Datenquellen realisiert.

13.2.2 Umsetzung

Durch die Verwendung von agilen Projektmanagement- und Softwareentwicklungsmethoden gab es keinen starren Architektur- und Softwareentwurf. Dennoch haben wir uns aufgrund des zu erwartenden hohen Komplexitätsgrades der Software für den Einsatz des Model-View-ViewModel Musters entschieden.

In einem ersten Planungsmeeting wurden, auf Basis der bis dato ermittelten Anforderungen, die benötigten Teilkomponenten der Software identifiziert und auf die MVVM Struktur abgebildet. Dabei wurde eine erste Leitarchitektur erstellt welche in Abbildung 13.2 zu sehen ist.

Nach der Erstellung der ersten Leitarchitektur wurde diese in Bezug auf die Kommunikation zwischen den einzelnen Komponenten überarbeitet und die Abbildung einzelner Softwarekomponenten auf die MVVM Struktur verfeinert. Im Laufe des Projektes wurde der entstandene Entwurf, wie in agilen Projekten vorgesehen, immer wieder angepasst um der aktuellen Funktionalität des Systems zu entsprechen. Eine gekürzte Fassung des Entwurfs findet sich in Abbildung 13.3. Im Folgenden werden die wesentlichen genutzten Module und ihre Einordnung in die MVVM Struktur näher erläutert.

View In der View-Schicht werden die Anzeigevarianten und Visualisierungen abgelegt. Somit erfolgt eine strikte Trennung zwischen der Anzeigekomponente und der Logik hinter der Anzeige.

Abbildung 13.2: Erster Entwurf der TOAD Architektur

ViewModel Die ViewModel-Schicht beinhaltet jeweils die Logiken der einzelnen View-Elemente. Somit besteht eine 1:1 Beziehung zwischen einem View-Element und einem ViewModel-Element.

Business Layer Im Business Layer werden Klassen gekapselt, die Daten verarbeiten und zwischen den View spezifischen und allgemeinen Klassen zur Datenhaltung austauschen. In diesem Projektkontext wurden die Funktionalitäten der Synchronisierung, der History sowie des Session-Managements in dieser Schicht gekapselt. Diese Funktionen werden aus den ViewModels aufgegriffen und haben jeweils direkt Zugriff auf das Model bzw. auf den Datalayer. Da zudem eine komplexe Verarbeitung der Objekte stattfindet, fand eine Einordnung im Business-Layer statt.

Model Im Model findet die Speicherung aller Daten zur Laufzeit des Programmes statt. So werden alle Analysedaten in dieser Schicht abgelegt. Sie dient somit als permanentes Repository welches über Bindings oder den Business Layer adressiert werden kann.

Datalayer Der Datalayer bietet den Zugang zu externen Daten. Es ist möglich Daten zu speichern(Sessions, Filter) oder neue Daten von Testfahrten in das Programm einzuladen. Dazu ist jeweils der XML-Parser zuständig, welcher die grundlegenden Dateien einlädt und die entsprechende Struktur des Prototypen überführt.

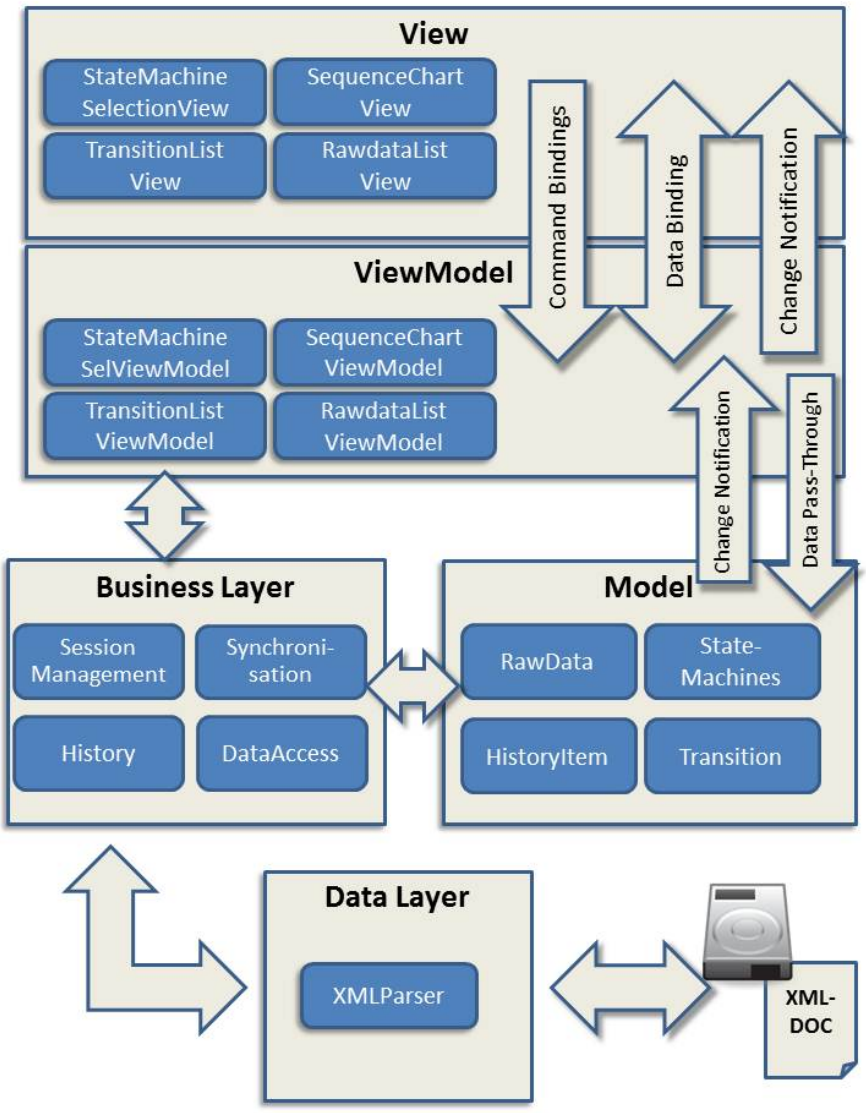


Abbildung 13.3: Gekürzter Entwurf der TOAD Architektur

13.2.3 Fazit

Das Model-View-ViewModel Muster erforderte zwar eine gewisse Einarbeitungszeit, erwies sich im Laufe des Projektes jedoch als hilfreiche Unterstützung bei der Strukturierung des Systems und der Beherrschung von dessen Komplexität.

Die strikte Trennung von View und ViewModel erlaubte eine einfache Anpassung der einzelnen Views ohne tiefgreifende Änderungen an der Programmlogik durchführen zu müssen. Dies erwies sich als eine große Hilfe bei der iterativen Entwicklung der einzelnen Visualisierungen, von den prototypischen Ideen hin zur finalen Version. Funktionen wie die Synchronisierung, History oder das Session Management konnten auf Basis der ViewModels realisiert werden ohne die konkrete Ausgestaltung der entsprechenden Views zu kennen.

Im Bereich der Touch-Verarbeitung stellte uns das MVVM Konzept jedoch vor große Herausforderungen. Touch-Events zur Manipulation der Darstellung können nicht direkt an die ViewModels weitergegeben werden, sondern müssen im Code-Behind abgefangen werden. Solche Manipulationen sind in der Regel abhängig vom aktuellen Zustand der View, wobei hier häufig nicht alle benötigten Parameter per Data Binding an das ViewModel übergeben werden konnten. Zur Lösung dieses Problems haben wir spezielle Commands entwickelt, auf die die jeweiligen Touch-Events gemappt wurden und denen alle zur Manipulation benötigten Parameter übergeben wurden.

Insgesamt gesehen profitierten die meisten Bereiche der Entwicklung jedoch von der Nutzung des MVVM Musters.

13.3 Module

In diesem Kapitel werden die Module, welche in dem Programm umgesetzt wurden, genauer erläutert.

13.3.1 Visualisierungen

Im Bereich der Datenvisualisierungen mussten um den Anforderungen zu entsprechen vier Visualisierungen implementiert werden (s. Kapitel 12.2). In den folgenden Abschnitten soll die Umsetzung der Visualisierungen erläutert werden.

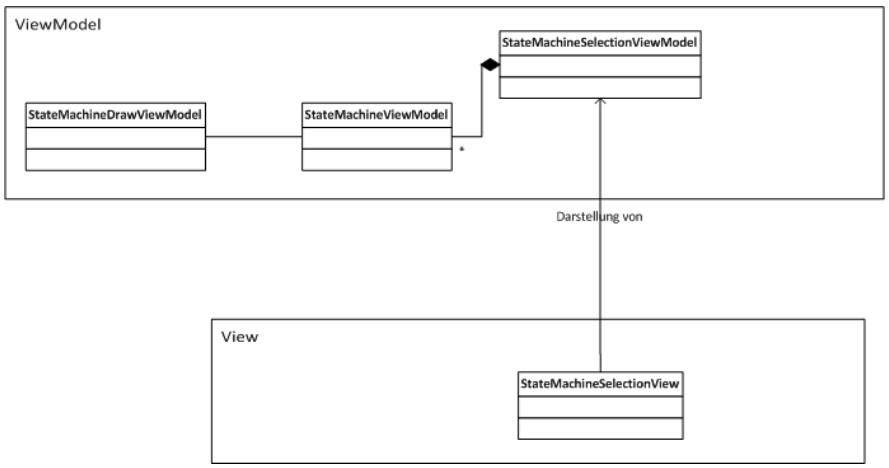


Abbildung 13.4: Klassenstruktur der Automatenansicht

Automatenansicht

Die Automatenansicht besteht aus der Visualisierung aller Automaten einer *Tracefile* sowie dem Umgang mit diesen. Die Funktionen gliedern sich dabei in das Darstellen der Automaten als farblich codierte Balken, dem scrollen in der Automatenansicht und die Aus- und Abwahl von Automaten. In Abbildung 13.4 sieht man den Aufbau der Automatenansicht auf Klassenebene.

In den nachfolgenden Abschnitten wird die farbliche Codierung sowie die Interaktion mit der Automatenauswahl auf Umsetzungsebene beschrieben.

Um die Automatenauswahl mit den farblich codierten Automaten zu füllen wird das aus der übergebenen Tracefile erstellte *Testrun*-Objekt durchlaufen wobei für jeden Automaten(*StateMachine*) ein Listeneintrag angelegt wird. Die einzelnen Listenelemente werden wiederum gefüllt in dem Bildrepräsentationen der Automaten erstellt und mittels *Binding* der View(*StateMachineListView*) zur Verfügung gestellt werden. Um die farbliche Codierung der Automaten zu ermöglichen wird ein *StateMachineDrawViewModel* erstellt, wobei ihm das Testrun-Objekt mit übergeben wird. Um die proportional korrekte Zeichnung der Automatenrepräsentation zu ermöglichen wird zunächst die Gesamtdauer der Testrun ermittelt. Um das dem *StateMachineListViewModel* übergebene *BitmapSource*-Objekt zu erstellen wird die TransitionsListe des Testrun-Objekts durchlaufen. Dabei wird für jede *Transition* ein Rechteck in der Farbe des gefundenen Zustandes gezeichnet (z.B. DEFECT = Rot). Die Breite des Rechteckes ergibt sich aus der Dauer der Transition in Bezug zur Gesamtdauer der Testrun, die Höhe ist dabei fest hinterlegt. Ist die Dauer einer Transition dabei zu gering um noch erkannt zu werden wird die Breite mit 2 Pixeln festgelegt.

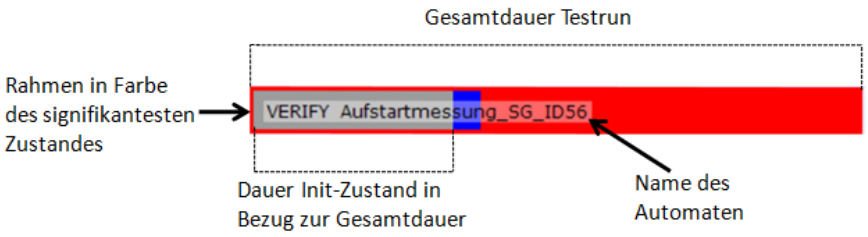


Abbildung 13.5: Automatendarstellung Bestandteile

Um den signifikantesten Zustand eines Automaten besser kenntlich zu machen wird um das Balkenbild ein Rahmen in der Farbe des signifikantesten Zustandes gelegt. Zum Abschluss folgt das Beschriften der Automatenrepräsentation durch zeichnen des Automatennamens auf der Bitmap. In Abbildung 13.5 sieht man den Aufbau einer erstellten Automatengrafik und ihre markierten Bestandteile.

Um eine weitere Exploration der Daten zu ermöglichen ist die Automatenliste eine touchfreundliche *SurfaceList* die ein einfaches scrollen innerhalb der Automatenliste mittels Scrollbalken am rechten und unteren Rand ermöglicht. Um eine detailliertere Darstellung eines Automaten durch ein *Zustandssequenzdiagramm* angezeigt zu bekommen sind die einzelnen Automaten in der Liste An- und Abwählbar. Dabei werden die selektierten StateMachines im *StateMachineSelectionViewModel* gehalten. In der View werden die als selektiert hinterlegten Automaten durch farbliche Hinterlegung gekennzeichnet. Pro angewählter StateMachine wird eine weitere Workbench geöffnet, die das dem ausgewählten Automaten entsprechende Zustandssequenzdiagramm anzeigt. Bei Abwahl eines Automaten wird auch das dazugehörige Zustandssequenzdiagramm wieder geschlossen.

Zustandssequenzdiagramm

Im Zustandssequenzdiagramm werden die Zustände eines Automaten sowie die Transitionen darin über die Zeit visualisiert. Die Umsetzung des Diagramms basiert dabei auf den Transitionsdaten des zu analysierenden Automaten. In Abbildung 13.6 ist die Klassenstruktur der Implementierung dargestellt, die sich nach MVVM in View und ViewModel trennen lässt.

Im *SequenceChartViewModel* werden in einem ersten Schritt alle Zustände des Automaten aus den Transitionsdaten extrahiert und entsprechende *StateViewModels* erstellt. Jeder Zustand kann über verschiedene Zeiträume aktiv sein. Diese Zeiträume werden durch *StateActiveTimeViewModels* repräsentiert, welche über einen Start- und

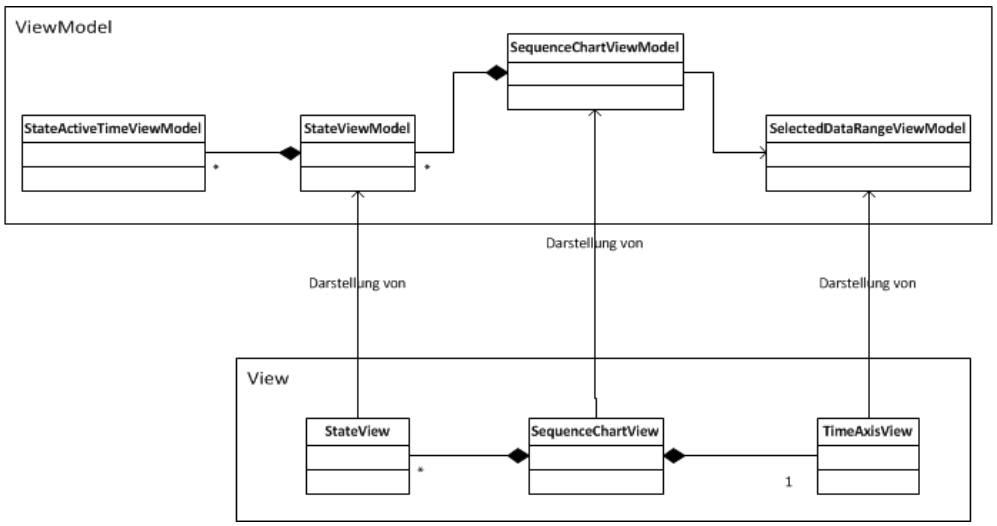


Abbildung 13.6: Klassenstruktur des Zustandssequenzdiagramms

einen Endzeitpunkt definiert sind. Diese Informationen werden ebenfalls aus den Transitionsdaten extrahiert. Der aktuell betrachtete Datenbereich des Zustandssequenzdiagramms ist im *SelectedDataRangeViewModel* repräsentiert. Initalisiert wird dieses mit der Gesamtlaufzeit des betrachteten Automaten.

Abbildung 13.7 zeigt wie die einzelnen ViewModels durch entsprechende Views dargestellt werden. Die Visualisierung des SequenceChartViewModels erfolgt über die *SequenceChartView*. Die SequenceView enthält zu jedem Zustand des Automaten eine *StateView*. Jede StateView visualisiert ein StateViewModel und dessen StateActiveTimeViewModels. Dabei werden nur StateActiveTimeViewModels dargestellt, die zeitlich mit dem SelectedDataRangeViewModel überlappen, das heißt im ausgewählten Datenbereich liegen. Diese StateActiveTimes werden durch farbige Balken visualisiert, deren Größe und Position in Abhängigkeit vom betrachteten Datenbereich und von der Größe der View berechnet wird. Diese Berechnung erfolgt u.a. mit Hilfe von *Value Convertern*, die Daten aus den ViewModels in direkt für die View nutzbare Daten umwandeln, zum Beispiel durch die Umwandlung von Zeitpunkten in Positionsdaten. Die farbigen Balken sind als Buttons realisiert, bei deren Berührung das Öffnen der Transitionsansicht durch die Synchronisierung (siehe Abschnitt 13.3.3) ausgelöst wird. Zwischen den Zuständen eines Automaten gibt es Transitionen, die einen Zustandswechsel repräsentieren und als Pfeile dargestellt werden. Da Transitionen über die Grenzen einer einzelnen StateView hinweggehen, werden sie direkt von der SequenceChartView in einer eigenen Ebene gezeichnet. Genauso wie bei der Visualisierung der StateActiveTimes helfen hierbei Value Converter bei der Berechnung der Größe und Position der Transitionspfeile. Alle visuellen Komponenten, die in den Views verwendet werden, haben gemeinsam, dass ihr Erscheinungsbild zu großen Teilen über Templates und Styles festgelegt wird. Diese sind in den XAML Dateien der

jeweiligen Views definiert.

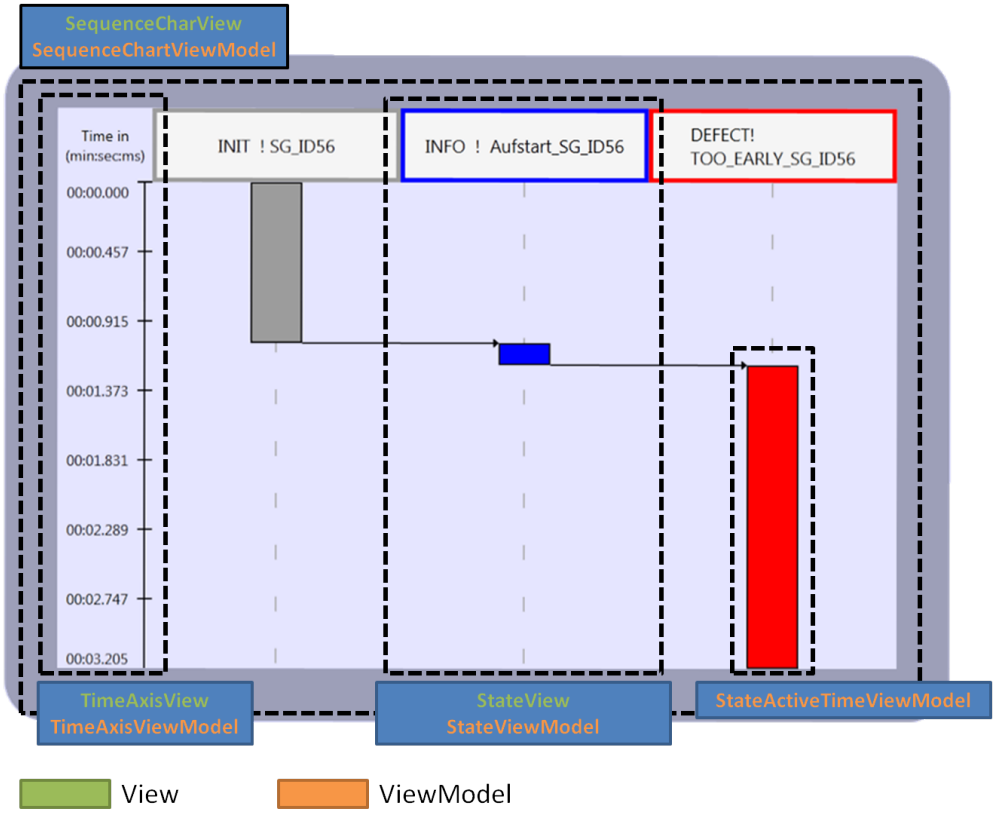


Abbildung 13.7: Zusammenhang von ViewModels und View im Zustandssequenzdiagramm

Das Zustandssequenzdiagramm unterstützt Zooming und Panning, das heißt das Vergrößern und Verkleinern des betrachteten Datenbereichs sowie das Verschieben des betrachteten Zeitfensters. Realisiert wird dies über eine entsprechende Veränderung des SelectedDataRangeViewModels. Die Interaktionen, die zur Manipulation führen, erfolgen, wie im Abschnitt zum Interaktionsdesign (siehe 12.4.2) beschrieben, über Gesten. Diese Gesten werden zunächst in der SequenceChartView abgefangen und dann in Parameter umgerechnet, die die Manipulation des betrachteten Datenbereichs beschreiben, wie zum Beispiel in einen Zoomfaktor. Über die Funktionen des SequenceChartViewModels wird die Manipulation im nächsten Schritt auf das SelectedDataRangeViewModel angewandt. Bei jeder Veränderung des SelectedDataRangeViewModels müssen die aktiven Zustandszeiten, die Transitionen und die Zeitachse neu gezeichnet werden, um dem aktuellen Zustand des ViewModels zu entsprechen. Eine Neuzeichnung dieser Komponenten ist auch bei Manipulationen der Workbench (siehe Abschnitt 12.4.1) erforderlich. Aus Performancegründen erfolgt diese Neuzeichnung allerdings erst nachdem die Manipulation der Workbench komplett abgeschlossen wurde.

Transitionsansicht

In der *TransitionListView* werden die Transitionen der gesamten Tracefile visualisiert. Die TransitionListView wird angezeigt, indem im Zustandssequenzdiagramm ein Zustand ausgewählt wird. Durch Auswahl eines Zustandes wird eine neue Workbench geöffnet, in der die Transitionen in Tabellenform dargestellt werden. Dabei wird dem *TransitionListViewModel* eine Liste mit allen Automaten in Form von *StateMachineViewModels* übergeben. Aus diesen werden alle TransitionListViewModels extrahiert und einer *ObservableCollection* hinzugefügt. Diese Liste wird danach den Zeitpunkten entsprechend mittels Sortieralgorithmus(*BubbleSort*-Verfahren) sortiert. Zusätzlich wird zur entsprechenden Transition im *DataGrid* gescrollt wobei sich die entsprechende Transition, sofern nicht am Anfang oder Ende der Liste, in der Mitte der Darstellung befindet. Das DataGrid wurde dabei in eine *SurfaceScrollView* Umgebung gesetzt um ein touchfreundliches Bedienen zu ermöglichen. Da die Anzahl der Elemente im DataGrid der Transitionsansicht eine relativ überschaubare Menge ist musste die Virtualisierung -im Gegensatz zur Rohdatenansicht- nicht aktiviert werden. Vorteil der nicht Virtualisierung war, dass ohne größeren Aufwand die oben genannte SurfaceScrollView-Umgebung eingesetzt werden konnte. Wie in den vorher beschriebenen Visualisierungen werden auch die Daten der TransitionListView im dazugehörigen ViewModel gehalten (TransitionListViewModel). Eine Sortierung nach den verschiedenen Spalten der TransitionListView lässt sich über eine *Tap*-Geste auf die jeweilige Spaltenüberschrift aufrufen. Folgende Spalten und Sortierungen stehen dabei zur Verfügung.

- Farbcodierung des Zielzustandes (Defect $\rightarrow$ Init oder Init $\rightarrow$ Defect)
- Zeitstempel (Auf- Absteigend Zeitpunkt)
- Startzustand (Auf- Absteigend Alphabetisch)
- Transitionsname (Auf- Absteigend Alphabetisch)
- Endzustand (Auf- Absteigend Alphabetisch)

Wird in der Transitionsansicht eine Transition durch berühren ausgewählt und sind Rohdaten im Testrun hinterlegt wird eine weitere Workbench geöffnet, die die Rohdatenansicht beinhaltet. Die Umsetzung dieser wird im nächsten Kapitel genauer vorgestellt.

Rohdatenansicht

In der *RawdataView* werden alle Rohdaten der gesamten Tracefile visualisiert. Die Rohdatenansicht basiert auf der *Rawdata*-Klasse aus dem Bereich Model. Für jedes

Rohdatum wird beim Einlesen der Tracefile ein neues Rawdata-Objekt erstellt. Die einzelnen Elemente aus dem XML-Rohdatum werden jeweils zu Attributen des Rawdata-Objekts geparsed. Aus jedem Rawdata-Objekt wird im nachfolgenden Schritt eine Instanz des *RawdataViewModel* erstellt. Das RawdataViewModel hat für alle Elemente, welche in jedem Rohdatum existieren (Bustype, Time, Data, ID, BusID und DLC) ein eigenes Attribut und darüber hinaus ein Attribut *XmlData* für alle weiteren, bustypabhängigen Elemente des Rohdatums. Aufbauend auf den RawdataViewModels wird für die Rohdatenansicht ein *RawdataListViewModel* erstellt. Das RawdataListViewModel beinhaltet alle RawdataViewModel der aktuellen Tracefile und speichert diese in einer *ObservableCollection*. Zustätzlich besitzt das RawdataListViewModel ein Attribut *SelectedTime*, welches durch den Konstruktur befüllt wird und die Zeit der, zuvor in der Transitionsansicht ausgewählten, Transition beinhaltet.

Die grafische Darstellung der Rohdatenansicht ist in *RawdataList.xaml* implementiert. Die RawdataList besteht aus einem *Datagrid*, welches dynamisch über ein Databinding aus dem RawdataListViewModel befüllt wird. Da eine Tracefile eine sehr große Menge an Rohdaten beinhalten kann, wurde für das Datagrid die Virtualisierung genutzt. Dies bedeutet, dass beim Aufruf der Ansicht nicht alle Rohdaten in die Ansicht geladen werden, sondern jeweils nur die aktuell anzuzeigenden Rohdaten. Beim Scrollen innerhalb der Ansicht werden die anzuzeigenden Rohdaten dann dynamisch nachgeladen. Der Vorteil der Virtualisierung ist, dass die Rohdatenansicht trotz ihrer sehr großen Datenmenge nach einer vergleichsweise kurzen Ladezeit angezeigt werden kann. Die Sortierung des Datagrids kann durch eine Tap-Geste auf eine Spaltenüberschrift des Datagrids durchgeführt werden. Es kann hierbei nach allen Spalten sortiert werden.

Die Rohdatenansicht wird erstmal angezeigt, wenn der Anwender in der Transitionsansicht eine Transition auswählt. Beim Aufruf der Rohdatenansicht scrollt die Anwendung automatisch zum Rohdatum mit dem zeitlich kürzesten Abstand zur zuvor gewählten Transition. Es wird zum zeitlich kürzesten Abstand gescrollt, weil in den Tracefiles nicht sichergestellt werden kann, dass zu jeder Transition ein Rohdatum mit dem exakt gleichen Zeitstempel existiert. Wenn der Anwender anschließend eine zweite Transition in der Transitionsansicht auswählt, wird keine neue Rohdatenansicht erstellt, sondern es wird nur zum entsprechenden Zeitpunkt gescrollt. Zu jeder Transitionsansicht existiert also maximal eine Rohdatenansicht.

13.3.2 Workbenchhandling

Das Workbenchhandling wurde mehrmals implementiert. Im ersten Teil dieses Abschnitts wird daher beschrieben, wie das Workbenchhandling ohne eine Trennung von ViewModel und Model stattgefunden hat, während im zweiten Teil beschrieben wird, wie diese Funktionalität in das MVVM-Muster umgesetzt wurde.

Workbenchhandling ohne MVVM

Im ersten Anlauf wurden die Möglichkeiten von WPF benutzt um Verschieben, Rotieren und Skalieren zu realisieren. Der Code dazu ist in mehreren Blogs im Internet zu finden, unter anderem auch im MSDN Blog *Lester's WPF\SL Blog*[les09]. Da wir im ersten Anlauf die Manipulation nicht nicht in Abhängigkeit vom Model bearbeiten mussten, haben die WPF Möglichkeiten vollkommen ausgereicht. Im Folgenden dazu die wichtigsten Befehle:

Um die Möglichkeit ein Objekt zu manipulieren zu aktivieren, müssen in dem Objekt, bei uns also in der Workbench, mindestens die in Quellcode 13.1 gezeigten Eigenschaften gesetzt sein.

```
<UserControl IsManipulationEnabled="True"
             ManipulationDelta="TransformOnManipulationDelta"
             ManipulationStarting="UserControlManipulationStarting"
             ManipulationInertiaStarting="UserControlManipulationInertiaStarting">
  <UserControl.RenderTransform>
        <MatrixTransform x:Name="MatrixTransform"/>
    </UserControl.RenderTransform>
  ...
</UserControl>
```

Listing 13.1: XAML-Eigenschaften eines UserControls zum Manipulieren.

UserControlManipulationStarting (Code 13.2) ist eine Funktion, in der Variablen gesetzt werden können, bevor die eigentliche Manipulation beginnt. In unserem Fall ist das nicht nötig, wodurch die Funktion recht kurz ausfällt.

```
private void UserControlManipulationStarting(object sender,
    ManipulationStartingEventArgs e) {
  e.Handled = true;
}
```

Listing 13.2: Die Funktion *UserControlManipulationStarting*.

TransformOnManipulationDelta ist die eigentliche Manipulations-Funktion. Wir haben mehrere Varianten ausprobiert, wie eine Workbench manipuliert werden kann. Eine Alternative zu der hier vorgestellten ist direkt auf der Matrix der Workbench zu arbeiten und so Rotation, Skalierung und Verschieben in einem Schritt durchzuführen. Dies hätte auch den Vorteil, dass man leichter den Mittelpunkt, um den diese Funktionen ausgeführt werden, ermitteln kann. Wir haben uns aber dazu entschieden die Schritte nacheinander abzuarbeiten, was unter anderem die Lesbarkeit des Codes erhöht,

da Standardfunktionen, statt mathematischer Formeln benutzt werden können. Den letztendlichen Ausschlag ob die Matrix benutzt werden sollte, oder nicht gab allerdings die Tatsache, dass es uns später nicht möglich war die Matrix nach der Trennung von ViewModel und View im ViewModel zu nutzen.

Die Funktionen `Scale`, `Rotate` und `Translate` skalieren, rotieren und verschieben die Workbench. In diesem ersten Ansatz werden für die tatsächliche Manipulation noch Funktionen der Matrix benutzt (`matrix.ScaleAt`, `matrix.RotateAt` und `matrix.Transform`). In der späteren Variante wurden andere Standard-Funktionen von WPF genutzt. Bei der Implementierung der drei Funktionen haben wir jeweils vor den eigentlichen Funktionen Fehleingaben möglichst abgefangen und nach den Funktionen überprüft, ob die Workbench in einem zulässigen Maße verändert wurde. So kann eine Workbench zum Beispiel nicht aus dem Anzeigebereich des Tisches geschoben werden oder größer als dieser Anzeigebereich dargestellt werden.

```
1  private void TransfromOnManipulationDelta(object sender, ManipulationDeltaEventArgs e)
       {
2    IEnumerator<IManipulator> enumerator = e.Manipulators.GetEnumerator();
3
4    // stop transformation, when leaving the workbench while touching
5    // [...]
6
7    // do translation
8    Point centre = Translate(e.DeltaManipulation.Translation);
9
10   // do rotation
11   Rotate(e.DeltaManipulation.Rotation, centre);
12
13   // scale
14   Scale(e.DeltaManipulation.Scale.X, centre);
15
16   e.Handled = true;
17 }
```

Listing 13.3: Die Funktion *TransformOnManipulationDelta*.

Um physikalisches Verhalten hinzuzufügen werden in der Funktion *OnManipulationInertiaStarting* Werte gesetzt. In unserem Fall haben wir festgelegt, wie lange eine Workbench sich noch weiter verschieben, skalieren oder rotieren soll, nachdem die Finger wieder vom Tisch genommen wurden. `DesiredDeceleration` gibt dabei die Geschwindigkeit in einer WPF-eigenen Einheit an und wurde von uns explorativ bestimmt.

```
1 private void OnManipulationInertiaStarting(object sender,
      ManipulationInertiaStartingEventArgs e) {
2   e.TranslationBehavior.DesiredDeceleration = 100d * 90d / (1000000d);
3   e.ExpansionBehavior.DesiredDeceleration = 0.1 * 90d / (1000000d);
4   e.RotationBehavior.DesiredDeceleration = 320 / (1000000d);
5 }
```

Listing 13.4: Die Funktion *OnManipulationInertiaStarting*.

Workbenchhandling mit MVVM

Im Laufe der Entwicklung ist die Anforderung aufgetreten die Position, Ausrichtung und Größe einer Workbench abhängig vom Model zu ändern. Beispielsweise um den Zustand einer Session nach dem Laden wiederherzustellen oder um eine Workbench neben einer bestehenden Workbench zu öffnen. Die Eigenschaften zur Ausrichtung, Größe und Position wurden daher in das ViewModel der Workbench übernommen und via *Data Binding* an die View gebunden. Die Funktionen `Scale`, `Rotate` und `Translate` konnten mit wenigen Änderungen ins ViewModel übernommen werden. Statt Standard-Funktionen von WPF aufzurufen werden die Werte für das Binding direkt bearbeitet. Da diese Werte nicht so einfach an die Matrix der View gebunden werden können, wurde statt der `MatrixTransform` eine Gruppe von Standard Tranformationen genutzt, wie im Code-Beispiel 13.5 gezeigt.

```
1  <UserControl.RenderTransform>
2    <TransformGroup>
3      <!-- Not needed anymore (see Canvas.Top and Canvas.Left Binding in TouchCanvasView
           ) -->
4      <!-- TranslateTransform x:Name="Translation"
5                           X="{Binding Path=TranslateX, Mode=TwoWay}"
6                           Y="{Binding Path=TranslateY, Mode=TwoWay}" / -->
7      <ScaleTransform x:Name="Scaling"
8                      CenterX="{Binding Path=ScaleCenterX, Mode=TwoWay}"
9                      CenterY="{Binding Path=ScaleCenterY, Mode=TwoWay}"
10                     ScaleX="{Binding Path=ScaleX, Mode=TwoWay}"
11                     ScaleY="{Binding Path=ScaleY, Mode=TwoWay}" />
12     <RotateTransform x:Name="Rotation"
13                      CenterX="{Binding Path=RotationCenterX, Mode=TwoWay}"
14                      CenterY="{Binding Path=RotationCenterY, Mode=TwoWay}"
15                      Angle="{Binding Path=RotationAngle, Mode=TwoWay}" />
16   </TransformGroup>
17 </UserControl.RenderTransform>
```

Listing 13.5: Neues *RenderTransform* der Workbench.

Eine Besonderheit bei dieser Variante ist, dass die Position der Workbench nicht ohne Probleme über die TranslateTransform verändert werden kann. Deshalb wird in der *TouchCanvasView*, also in dem Bereich, in der die alle Workbenches liegen, `Canvas.Top`

und `Canvas.Left` gebunden um die Workbenches zu verschieben.

Da nur die View auf Touch-Ereignisse reagieren kann, war es die größte Herausforderung bei der MVVM konformen Umsetzung das ViewModel zu benachrichtigen, sobald die Workbench verschoben werden soll. Die Rückrichtung ist ja bereits über das Data Binding möglich. Selbst wenn es möglich wäre das ViewModel zu benachrichtigen, wäre unbekannt, wie weit verschoben, wie viel vergrößert oder um wie viel Grad gedreht werden soll. Zur Lösung des Problems haben wir eine eigenes Command erstellt, wie es in Abschnitt 13.2.1 definiert wird. Das in Abbildung 13.8 dargestellte `ManipulateWorkbenchCommand` wird im ViewModel mit den Funktionen für Verschieben, Skalieren und Rotieren instantiiert und über ein Data Binding and die View gebunden. Tritt ein Touch-Event auf, kann das Kommando über das Binding in der View gefunden werden und mit der Funktion `Execute` ausgeführt werden. Durch das Interface `ICommand`, das hierbei implemeniert wird, ist vorgegeben, dass `Execute` jeden beliebigen Parameter übernimmt. Allerdings kann mit `CanExecute` prüfen, ob der Parameter gültig ist. In unserem Fall werden nur vierstellige Arrays als Parameter übernommen. Das erste Feld enthält dabei einen Vektor der angibt wohin die Workbench bewegt werden soll. Das zweite Feld enthält einen Vector, der angibt wie weit die Workbench in X- und Y-Richtung skaliert werden soll. Im dritten Feld ist ein `double`-Wert gesetzt, der die Rotation angibt. Im letzten Feld sind die Ausmaße des aktuellen Anzeigebereichs des Tisches als `Rect`-Wert gespeichert.

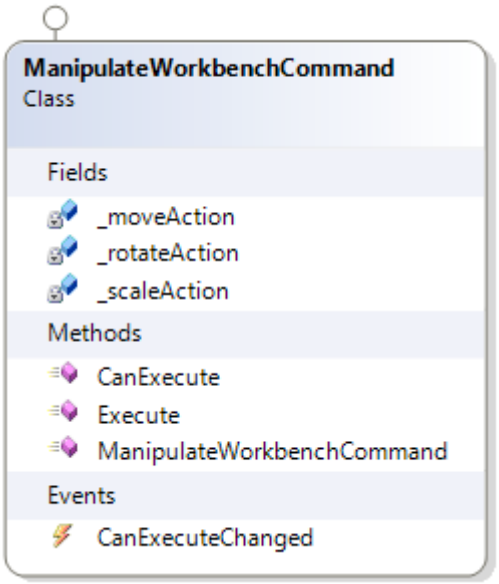


Abbildung 13.8: Die Klasse *ManipulateWorkbenchCommand*

Ein Aufruf von `Execute` mit einem korrekten Parameter führt schließlich dazu, dass die zuvor im Viewmodel gesetzten Funktionen mit den richtigen Werten ausgeführt werden, damit die Werte für das Data Binding verändert werden und schließlich die Workbench in der View verändert wurde. In Abbildung 13.9 wird der bereits beschriebene Prozess konkret für eine Rotation dargestellt. Der Ablauf zeigt dabei, wie die Daten durch das System gereicht werden, welche Funktionen tatsächlich aufgerufen werden wurde hier nicht berücksichtigt.

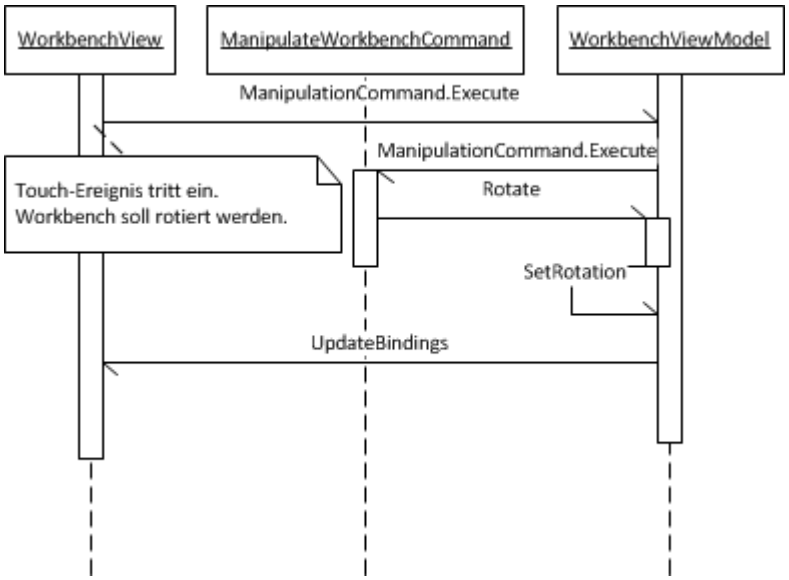


Abbildung 13.9: Ein stark vereinfachtes Sequenzdiagramm zur Durchführung einer Rotation.

13.3.3 Synchronisierung

Die Synchronisierung teilt sich in die Verknüpfung aller einzelnen Workbenches sowie in spezielle Synchronisierungsfunktionen auf. Diese speziellen Funktionen beinhalten das Kopieren und das Spiegeln von Workbenches. Die beiden Bereiche werden folgend vorgestellt.

Verknüpfung Workbenches

Die Funktionen der Synchronisierung werden immer genutzt wenn eine Workbench mit einer anderen Verknüpft wird. Dies erfolgt jeweils im Analyseablauf, wenn ein Element ausgewählt wird und sich eine neue Ansicht in einer weiteren Workbench öffnet. Die jeweiligen Fälle werden folgend genauer beschrieben.

Automatenansicht → öffnen Zustandssequenzdiagramm Sofern in der Automatenansicht ein Automat angewählt oder abgewählt wird, erfolgt der Aufruf der Synchronisierung.

Bei einer Anwahl eines Automaten wird ein neues ViewModel mit dem ausgewählten Automaten erzeugt und an die Synchronisierung übergeben. Diese erzeugt nun eine neue Workbench und setzt bei dieser die Anzeige auf das neu erzeugte Zustandssequenzdiagramm. Zudem wird ein Eintrag in der History erzeugt um diesen Schritt evtl. rückgängig machen zu können.

Bei einer Abwahl eines Automaten wird die zu löschende Workbench(mit dem Zustandssequenzdiagramm) ermittelt und mit dem abgewählten Automaten an die Synchronisierung übergeben. Nach dem Entfernen der Workbench wird ein Eintrag der History hinzugefügt um diesen Schritt rückgängig machen zu können.

Zustandssequenzdiagramm → öffnen Transitionsliste Druch die Anwahl oder Abwahl eines Zustandes im Zustandssequenzdiagramm erfolgt der Aufruf der Synchronisation, welche eine neue Transitionsliste erstellt oder schließt.

Bei der Anwahl/Abwahl eines Zustandes wird dieser an die Synchronisation übergeben. Diese prüft nun ob bereits eine synchronisierte Workbench besteht, die mit diesem Zustand verknüpft ist. Wenn dies der Fall ist wird die Workbench entfernt und ein Eintrag in der History hinzugefügt. Dies stellt das Abwählen eines Zustandes dar. Wenn keine korrespondierende Workbench zu dem Zustand besteht, wird eine neue Workbench erstellt. Auf dieser Workbench wird eine Transitionsliste erstellt, welche den Datenbereich darstellt, in dem sich der Zustand befindet. Abschließend wird ein Eintrag in der History hinzugefügt.

Transitionsliste → öffnen Rohdatenansicht Durch die Anwahl eines Elementes in der Transitionsliste wird eine Rohdatenansicht erstellt. Es wird jedoch nicht, wie bei den anderen Ansichten, durch eine Abwahl eines Elementes die Rohdatenansicht geschlossen. Dies begründet sich in der Größe der Rohdatentabellen. So wäre der Aufwand zu groß eine Tabelle mit ca. 50.000 Zeilen oft neu zu laden. Bei einer Neuselektion eines Elements wird in der Rohdatenansicht zu der jeweiligen Stelle gescrollt.

Bei der Anwahl eines Elementes wird die Synchronisierung ausgeführt. Dieser wird das selektierte Element aus der Transitionsliste übergeben. Sofern schon eine

Rohdatenansicht mit der Transitionsliste verbunden ist, wird diese an den jeweiligen Zeitpunkt gescrollt. Wenn noch keine Rohdatenansicht im Analyseablauf besteht, wird eine neue Workbench erstellt und in dieser die Rohdatenansicht angezeigt. Es wird dabei auch zu dem Zeitpunkt von der ausgewählten Transition gescrollt. Abschließend wird ein Eintrag in der History erstellt.

spezielle Synchronisierungsfunktionen

Die speziellen Synchronisierungsfunktionen setzen sich aus *Kopieren* und *Spiegeln* zusammen.
Beim Kopieren steht zudem die Option, dass alle Workbenches, welche in der Analysehierarchie tiefer stehen, mit kopiert werden. Somit ist es möglich einen gesamten Analysestrang zu kopieren und einem anderen Analysten bereit zu stellen.
Beim Spiegeln kann jeweils nur eine Workbench genutzt werden. Diese bekommt nach der Spiegelung eine weitere Workbench, welche exakt die gleichen Dinge anzeigt, wie die Ursprungsworkbench. Dies wird dadurch erreicht, dass die ViewModels beider Workbenches auf ein Objekt referenzieren. Der genaue Ablauf der Synchronisierung wird folgend genauer erläutert.

Kopieren Beim Kopieren einer Workbench gibt es zwei unterschiedliche Aufrufoptionen für die Synchronisierung. Es wird unterschieden zwischen dem Kopieren von einer Workbench und einer Workbench mit angeschlossenem Analysepfad.

Ohne Analysepfad Wenn die Option *ohne Analysepfad* aufgerufen wird, dann wird die ausgewählte Workbench sowie das ViewModel der Quellworkbench an die Synchronisierung übergeben. In einem ersten Schritt wird eine neue Workbench erstellt und dieser werden die Testrundaten übergeben. Anschließend müssen die anzeigespezifischen Einstellungen kopiert werden. Dazu erfolgt eine Fallunterscheidung anhand des Anzeigetyps der Quellworkbench.

Quellworkbench = Automatenansicht Es wird die Automatenansicht geladen. Da keine Elemente selektiert sind, müssen keine weiteren Einstellungen mit den Daten vorgenommen werden. Im letzten Schritt werden die Scrollbalken für X- und Y-Position noch anhand der Quellworkbench gesetzt.

Quellworkbench = Zustandssequenzdiagramm Es wird aus dem Quellzustandssequenzdiagramm der Automat geladen, welcher angezeigt wird. Anschließend wird ein neues Zustandssequenzdiagramm mit den Automatendaten erzeugt. Dieses wird der neuen Workbench zugewiesen und angezeigt. Abschließend

müssen noch *Listener* auf dem neuen Zustandssequenzdiagramm registriert werden, damit dieses auf Änderungen reagieren kann. Dies muss nur beim Zustandssequenzdiagramm durchgeführt werden, da dies selbst entwickelt wurde und auch von der Synchronisierungsklasse mit betreut wird.

Quellworkbench = Transitionsliste Es wird die Transitionsliste in der neuen Workbench geladen. Anschließend müssen, analog zur Automatenansicht, die Scrollbalken für die X- und Y-Position anhand der Quellworkbench gesetzt werden.

Quellworkbench = Rohdatenansicht Es wird die Rohdatenansicht in der neuen Workbench geladen. Anschließend wird die Scrollposition anhand der ausgewählten Zeit in der Quellworkbench ermittelt und an die neue Workbench übertragen. So erfolgt das Scrollen auf die gleiche Position.

Mit Analysepfad Wenn der Analysepfad mit kopiert werden soll, werden alle Workbenches, die in der Analysehierachie unter der zu kopierenden sind, mit kopiert. Wenn es keine synchronisierten Workbenches gibt, wird die Funktion *ohne Analysepfad* aufgerufen.

Die Funktion wird rekursiv für jede Analysehierarchiestufe durchlaufen und führt jeweils, in Abhängigkeit zum Anzeigetyp, die anzeigespezifischen Einstellungen durch. An die Funktion müssen die Quellworkbench, die Zielworkbench sowie die Quellanzeige übergeben werden. Der Ablauf des Kopieren orientiert sich, wie beim Kopieren ohne Analysepfad, anhand der Anzeigetypen und wird folgend genauer erläutert.

Die Folgende Abbildung 13.10 zeigt den Ablauf beim Kopieren mit Analysepfad. Es werden im ersten Schritt die jeweilige Workbench kopiert und in einem 2. Schritt die jeweiligen Einstellungen gesetzt. Anschließend wird die nächste Hierachiestufe aufgerufen und der Ablauf wiederholt.
Die folgende Auflistung zeigt den Ablauf für jeden Anzeigetyp pro Durchlauf der Synchronisierung.

Quellworkbench = Automatenansicht Die Automatenansicht wird in der Zielworkbench geladen. Anschließend werden die Scrollbalken anhand der X- und Y-Positionen der Quellworkbench gesetzt und die selektierten Automaten übertragen. Durch das setzten der selektierten Automaten werden automatisch neue Zustandssequenzdiagramme in den entsprechenden Workbenches erzeugt. Falls die Zustandssequenzdiagramme noch weitere Verknüpfungen zu Transitionslisten haben, wird ein weiterer Durchlauf der Synchronisierung gestartet mit dem Zustandssequenzdiagramm als Quellworkbench.

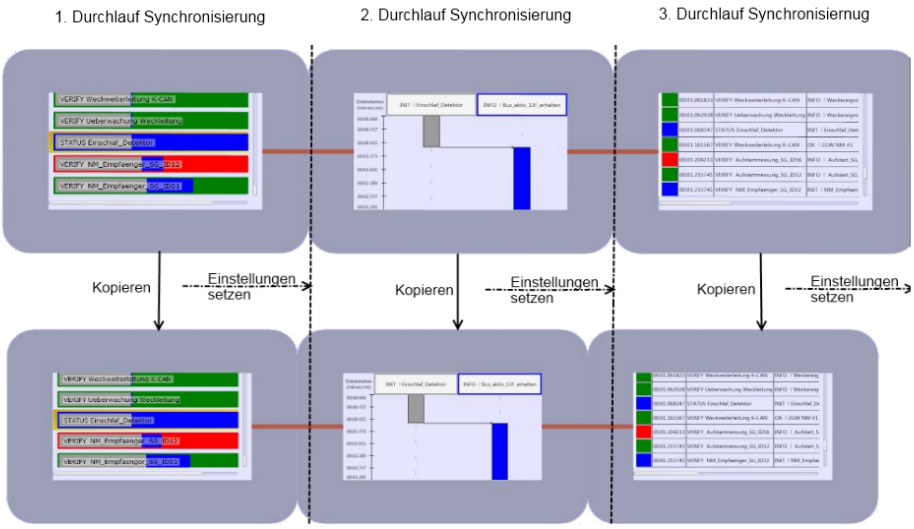


Abbildung 13.10: Ablauf Synchronisierung

Quellworkbench = Zustandssequenzdiagramm Sofern ein Zustandssequenzdiagramm der Einstieg des Kopierens war, wird diese Ansicht in der Zielworkbench erzeugt. Wenn das Zustandssequenzdiagramm Teil des Analyseablaufes ist, wird diese im Zielanalyseablauf ausgewählt. Anschließend werden die entsprechenden Zustände anhand der Quellworkbench angewählt. Somit entstehen korrespondierend zum Quellanalyseablauf neue Workbenches mit den Transitionslisten. Falls diese noch weiter verknüpft sind mit Rohdatenansichten, wird ein weiterer Durchlauf der Synchronisation gestartet mit der Transitionsliste als Quellworkbench.

Quellworkbench = Transitionsliste Sofern die Transitionsliste der Einstieg des Kopierens war, wird diese Ansicht in der Zielworkbench erzeugt. Wenn die Transitionsliste Teil des Analyseablaufs ist, wird diese im Zielanalyseablauf ausgewählt. Anschließend werden die entsprechenden Zustände anhand der Quellworkbench angewählt und die Scrollbalken anhand der X- und Y-Positionen der Quelltransitionsliste positioniert. Somit entstehend korrespondierend zum Quellanalyseablauf neue Workbenches mit den Rohdatenansichten. Die Rohdatenansichten können keine weiteren Verknüpfungen im Analyseablauf haben. Somit erfolgt kein weiterer Ablauf der Synchronisierung.

Quellworkbench = Rohdatenansicht Es muss keine Aktion durchgeführt werden, da keine Elemente auf der Rohdatenansicht selektiert werden können.

Spiegeln Das Spiegeln stellt eine Möglichkeit der Synchronisierung bereit, bei der nicht nur der Dateninhalt kopiert wird, sondern auch die Interaktionen, die mit den Daten erfolgen. Dabei ist es jedoch nur möglich das Spiegeln mit einer Quellworkbench

zu nutzen. Es ist nicht möglich einen Analysepfad mit zu spiegeln. Zudem wird die Funktion spiegeln nicht für eine Rohdatenansicht angeboten. Für die anderen Visualisierungen wird die Umsetzung folgend erläutert.

Automatenauswahl Bei der Automatenauswahl wird das ViewModel der Zielworkbench auf das der Quellworkbench gesetzt. Somit werden alle Interaktionen auf beiden Workbenches gleich dargestellt. Dies kann erreicht werden durch die lose Kopplung zwischen View und ViewModel. Dazu werden die Daten der Scrollbalken (X- und Y-Position) ins ViewModel kopiert und in beiden Views aktualisiert. Dies erfolgt analog bei den angewählten Automaten.

Zustandssequenzdiagramm Beim Zustandssequenzdiagramm wird das ViewModel der Zielworkbench auf das der Quellworkbench gesetzt. Somit werden alle Interaktionen auf beiden Workbenches gleich umgesetzt. Dies wird auch hier erreicht durch die lose Kopplung zwischen View und ViewModel. Die Zoom- und Scroll-Daten werden im ViewModel gespeichert und stehen somit beiden Views zur Verfügung.

Transitionsliste Die Umsetzung für die Transitionsliste erfolgt analog zur Umsetzung der Automatenansicht.

13.3.4 History

Die *History* repräsentiert den Gesamtablauf eines Analysepfades ausgehend von der Datenauswahl (s. Abb. 13.11). Dabei werden die wichtigsten Analyseschritte gespeichert und können durch Ausführen von Redo/Undo wieder angezeigt werden.

Ein Historyschritt wird angelegt, wenn der Anwender eine der Hauptansichten öffnet oder schließt. Die History speichert diese Schritte linear in chronologischer Reihenfolge. Wenn der Anwender per Geste eine neue Workbench erstellt, wird eine eigene History angelegt, an deren Wurzel immer der Datenauswahlbildschirm steht. Im Anschluss daran werden folgende Aktionen in die History aufgenommen:

- Laden einer XML-Datei,
- An- und Abwählen eines Automaten in der Automatenansicht,
- An- und Abwählen eines Zustandes im Zustandssequenzdiagramm,
- An- und Abwählen einer Transition in der Transitionsliste,
- Schließen einer Workbench über das Menü.

Die History kann über das Menü für jede Workbench eines Analyseablaufes separat angezeigt werden. Dabei sind jedoch alle Historys eines Ablaufes identisch. So macht es keinen Unterschied, ob ein Redo/Undo über die History eines Zustandssequenzdiagrammes oder der Rohdatenansicht ausgeführt wird. Das Schließen der Hauptworkbench kann nicht rückgängig gemacht werden, da dann der gesamte Analyseablauf verworfen wird.

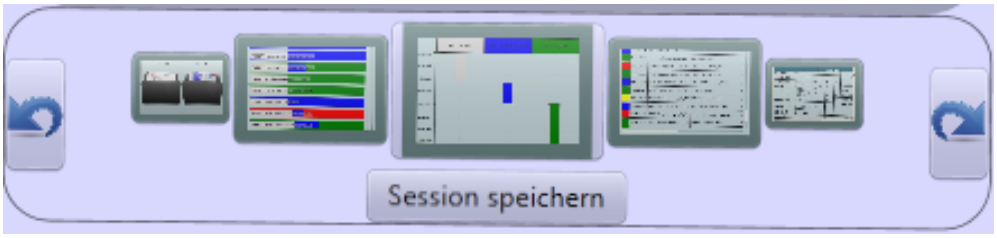


Abbildung 13.11: Historymenü und -anzeige

13.3.5 Session Management

Das Session Management realisiert Funktionen zum Speichern und Laden eines Analyseablaufes. Der Ladevorgang geschieht analog zum *Kopieren mit Analysepfad* der Synchronisierung (siehe 13.3.3). Beim Speichern werden die dazu benötigten Eigenschaften in einer XML Datei gespeichert und beim Laden entsprechend daraus umgewandelt. Beim Aufruf der Speicherfunktion aus einer Workbench heraus wird der gesamte Analysepfad, beginnend bei der Automatenauswahl, gespeichert. Die dabei erstellte XML Datei wird mit einem Zeitstempel im Home-Verzeichnis angemeldeten Benutzers gespeichert.

Listing 13.6 zeigt den Aufbau einer Session XML Datei des beispielhaften Analysepfads aus Abbildung 13.12. Das *MainWorkbench* Element repräsentiert die Workbench der Automatenauswahl, in der ein bestimmtes Logfile geladen wurde. In der Automatenauswahl können mehrere Automaten selektiert sein, wobei zu jedem selektierten Automaten ein Zustandssequenzdiagramm geöffnet ist. Dies wird über das Element *SelectedStateMachines* mit seinen Kindelementen *SelectedStateMachine* abgebildet. In einem Zustandssequenzdiagramm können mehrere aktive Zustandszeiten aktiviert sein wozu jeweils eine Transitionsliste gehört. Dies wird durch die *SelectedStateActiveTime* Elemente repräsentiert. In der Transitionsliste kann genau eine Transition selektiert sein, so dass die Transitionsliste mit genau einer Rohdatenansicht verknüpft ist, was durch das *SelectedTransition*-Objekt ausgedrückt wird. Eigenschaften, die die einzelnen Elemente identifizieren, sind als Attribute der jeweiligen Elemente realisiert.

```
<SavedSession>
  <MainWorkbench ID="1" Testrunxml="Logfile.xml">
    <SelectedStateMachines>

        <SelectedStateMachine Name="VERIFY Aufstartmessung_SG_ID08">
            <SelectedStateActiveTime Statename="INFO ! Aufstart_SG_ID08" Starttime
                ="00:01:060902" Endtime="00:03:205763" />
        </SelectedStateMachine>

        <SelectedStateMachine Name="VERIFY  Aufstartmessung_SG_ID56">
            <SelectedStateActiveTime Statename="INIT ! SG_ID56" Starttime
                ="00:00:000000" Endtime="00:01:060902">
                <SelectedTransition OriginalState="INIT ! SG_ID4D" DestinationState="
                    INFO ! Aufstart_SG_ID4D" Time="00:01.060902" />
            </SelectedStateActiveTime>

            <SelectedStateActiveTime Statename="DEFECT ! TOO_EARLY_SG_ID56" Starttime
                ="00:01:204213" Endtime="00:03:205763">
                <SelectedTransition OriginalState="INFO ! Aufstart_SG_ID12"
                    DestinationState="OK ! SG_ID12" Time="00:01.215741" />
            </SelectedStateActiveTime>
        </SelectedStateMachine>

    </SelectedStateMachines>
  </MainWorkbench>
</SavedSession>
```

Listing 13.6: XML Format der gespeicherten Session des Analysepfads aus Abbildung 13.12

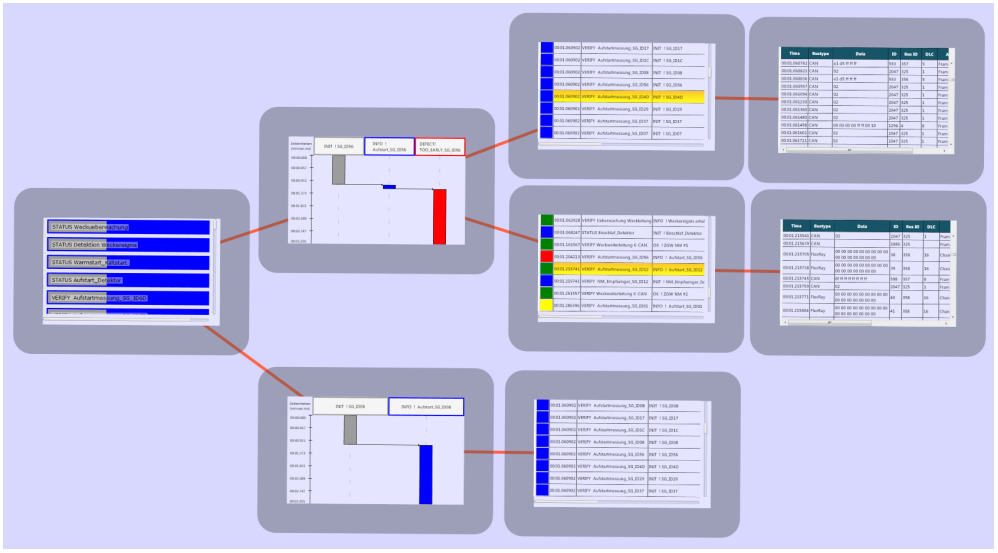


Abbildung 13.12: Analysepfad der gespeicherten Session aus Listing 13.6

13.3.6 Filter

Das Filtern teilt sich in zwei Bereiche auf: dem Filter setzen sowie dem Anwenden eines zuvor definierten Filters. Durch diese Funktionalität ist es möglich, interessante Datenbereiche festzulegen und dieses Wissen in späteren Analysen anzuwenden.

Filter setzen

Bevor Filter angewendet werden können, muss mindestens ein Filter zuvor definiert worden sein. Diese Funktionalität kann innerhalb des Zustandssequenzdiagrammes über das Marking Menu aufgerufen werden. Wird diese Funktion aufgerufen, so wird ein *BookmarkSelectionlistViewModel* sowie die zugehörige *BookmarkSelectionlistView* erstellt. Die View enthält eine SurfaceListBox, die mit einem passendem Property vom Typ ObservableCollection des ViewModels befüllt wird. Die Befüllung der Property geschieht durch eine Methode, die im Konstruktor aufgerufen wird. Das erste Element der ObservableCollection ist ein festgelegtes Element, um später auch das Abbrechen ermöglichen zu können. Als nächstes wird der Name des aktuellen Automaten als Auswahl angeboten, gefolgt von der Abfolge aller Zustände. Im Anschluss werden noch die Namen der verschiedenen Zustände zur Verfügung gestellt, sowie die Auswahl eines ganz bestimmten Zustandsüberganges. Wird später einer dieser Einträge durch einen Benutzer ausgewählt, wird die Methode zum Setzen eines Filters aufgerufen. Innerhalb dieser Methode wird geprüft ob das Element zum Abbrechen ausgewählt wurde und gegebenenfalls in das Zustandssequenzdiagramm zurückgesprungen. Andernfalls wird die Auswahl des Benutzers durch ein spezielles Schema in eine Textdatei geschrieben die sich im TOAD-Verzeichnis unter App-Data befindet. Je nach Auswahl eines Filters werden dessen interne Nummer sowie weitere kontextabhängige Informationen getrennt durch „|“ gespeichert. Der Nutzer erhält noch eine kurze visuelle Bestätigung, die durch einen drei-sekündigen Timer gesteuert wird, gefolgt von einem Wechsel zurück in das Zustandssequenzdiagramm.

Filter anwenden

Ein Filter kann innerhalb der Automatenansicht verwendet werden. Der Aufruf erfolgt auch hier durch das Marking Menu und es wird ein *BookmarkChooselistViewModel* sowie die zugehörige *BookmarkChooselistView* erstellt. Die View enthält ebenfalls eine SurfaceListBox, die mit einem passendem Property vom Typ ObservableCollection des ViewModels befüllt wird. Analog zum Filter setzen geschieht die Befüllung des Property durch eine Methode, die im Konstruktor aufgerufen wird. Das erste Element der ObservableCollection enthält ebenfalls ein festgelegtes Element, um später das Anzeigen der ursprünglichen Automatenansicht (durch Aufhebung aller Filter) zu ermöglichen.

Die restlichen Elemente des Propertys werden aus der Textdatei des TOAD-Ordners gelesen. In der SurfaceListBox können mehrere Elemente ausgewählt werden. Wird das Element zum Abbrechen bzw. Zurücksetzen ausgewählt so wird die Automatenansicht in der ursprünglichen Form geöffnet, so dass keiner der zuvor angewandten Filter mehr gesetzt ist. Bei Auswahl eines oder mehrerer Elemente sowie deren Bestätigung wird in das StateMachineSelectionViewModel gewechselt und es werden die ausgewählten Filter mit Hilfe eines Arrays an eine Methode dort übergeben. Die Methode durchläuft jeden der im Testrun vorhandenen Automaten und prüft diese auf die mitgegebenen Filter. Stimmen diese überein, so werden diese Automaten einer temporären Liste übergeben. Nachdem alle Automaten durchlaufen wurden wird die temporär erstellte Liste an das Property der Automatenansicht übergeben, welche üblicherweise alle Automaten aufweist, nun aber nur noch die durch die Filter eingeschränkte Auswahl.

13.3.7 Marking Menu

Um eine möglichst einfache und schnell zu nutzende Interaktionsmöglichkeit zu bieten um Grundfunktionalitäten aufzuführen (zum Beispiel Workbench schließen oder Programm beenden) oder Funktionalitäten die sich auf bestimmte Views beziehen (zum Beispiel Filter setzen oder Filter anwenden) wurde das Marking Menü (MM) entwickelt. Hierbei wurde darauf geachtet, dass dieses möglichst unabhängig vom restlichen TOAD Projekt zu halten war um die Einbindung in andere Projekte zu ermöglichen.

Das MM an sich besteht aus zwei Klassen der eigentlichen MM-Funktionalität im *Menu*-Objekt und der Haltung der Menüeinträge als *Item*.

Das Einbinden des Menüs erfolgte in der *WorkbenchView*. Hier wurde eine *PressAnd-Hold*-Geste realisiert. Dafür wurde ein *MarkingMenuTimer* implementiert der von einem *DispatcherTimer* erbt. Zusätzlich kann dem Timer die Position eines TouchEvent übergeben werden. Ist der Timer abgelaufen (nach der letzten Evaluation ist das nach 500ms der Fall) wird das MM eingeblendet. Um eine Überschneidung mit den WorkbenchHandling-Events zu vermeiden wird die gesamte Transformations-Funktionalität der Workbench deaktiviert während ihr Menü angezeigt wird. Über ein TouchUp Ereignis wird erkannt, wann die Manipulation der Workbench wieder aktiviert wird.

Damit das Marking Menu mit Items gefüllt wird existiert die abstrakte Klasse *Screen* welche die Methode *GetScreenSpecificMenuItems* zur Verfügung stellt. Sollen zusätzliche Items im Menü eines Screens (z.B. des Automatenauswahl-Screen) hinterlegt werden, kann diese Methode überschrieben werden. In Abbildung 13.13 ist der einfache Ablauf vom Tap bis zum Erscheinen des Menüs als Skizze dargestellt.

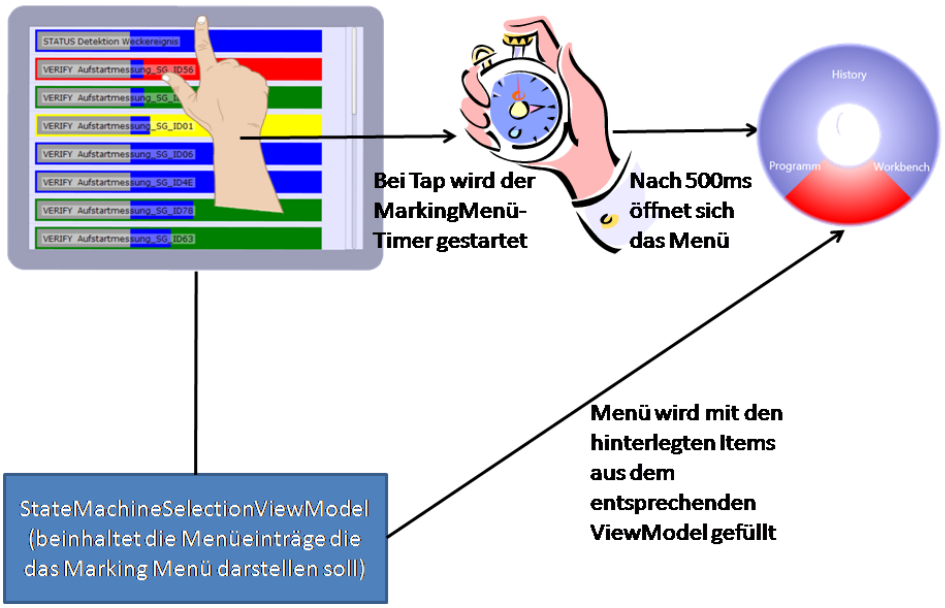


Abbildung 13.13: Vereinfachter Ablauf der Marking Menu Darstellung

13.4 KnoVA

Wissensextraktion ist bei der Analyse von Fahrzeugdaten ein sehr wichtiges Feld, da das Auffinden von Anomalien in Busdaten sehr stark abhängig vom Wissensstand des Analysten ist. Die Extraktion, Speicherung und Wiederanwendung von gesammelten Wissen ist daher ein wichtiger Aspekt in der Anwendung. Die Anwendung konnte mit Hilfe des KnoVA Referenzmodells um die Aspekte der Wissensextraktion erweitert werden. Eine grundlegende Vorstellung des KnoVA Modells wurde bereits in Kapitel 6 durchgeführt. Im folgenden werden die wissensbasierten Funktionen des TOAD-Systems zusammen gefasst und die Realisierung auf Basis des KnoVA Referenzmodells beschrieben.

13.4.1 Implementierung des KnoVA Referenzmodells

Die, in der Anwendung existierenden, Systemelemente werden über verschiedene Eigenschaften beschrieben. Diese Eigenschaften teilen sich auf in elementspezifische sowie in elementunspezifische Eigenschaften. Die unspezifischen Eigenschaften überschneiden sich bei den jeweiligen Elementen und bilden so den Ansatz für das

Referenzmodell.

Für die Wiederverwendbarkeit des Wissens aus Analyseabläufen ist ein standardisierter Aufbau der Visualisierungen nötig. Diese Visualisierungen, im Programmkontext als View bezeichnet, stellen einen spezifischen Zustand dar, welcher bestimmte Eigenschaften besitzt und eine Abstraktionsebene auf die Daten darstellt.

Der Zustand einer View wird durch die eigenen Eigenschaften beschrieben und ist somit eindeutig. Funktionen wie beispielsweise die History oder die Synchronisierung bauen auf diesem Konzept auf, da sie die parametrisierten Views nutzen und so die jeweiligen Zustände wieder herstellen können.

Da die Anwendung vier spezifische Visualisierungen nutzt, wurde ein Referenzmodell eingeführt, welches die Parametrisierung vereinheitlicht. Dies wurde durch die Vorgaben des MVVM-Modells unterstützt. Eine detaillierte Darstellung des MVVM-Modells erfolgt in Kapitel 13.2. Für die Umsetzung des KnoVA Referenzmodells ist die eindeutige Verbindung zwischen View und ViewModel, sowie deren standardisierte Kommunikation von Bedeutung. Durch diese Vorgaben konnte ein standardisierter Rahmen geschaffen werden, welcher das Referenzmodell abbildet. Innerhalb dieses Rahmens befinden sich die einzelnen Views, welche über spezifische Parameter verfügen. Die anschließende Abbildung 13.14 verdeutlicht dies nochmal. Auf dieser sind einerseits die visualisierungsspezifischen Parameter der, in der Anwendung umgesetzten, Views dargestellt sowie die Standardparameter, welche alle Views nutzen. Die Abbildung dient somit der Übersicht über alle Parameter, welche im Referenzmodell genutzt werden.

Die Standardparameter sowie die visualisierungsspezifischen Parameter werden im Anschluss genauer erläutert.

Standardparameter Standardparameter stehen bei allen Views zur Verfügung und stellen das grundlegende Modell zur Verfügung. Diese Daten sind unabhängig von der angezeigten Visualisierung und parametrisieren alle viewunspezifischen Eigenschaften. Die technische Realisierung erfolgt dabei durch ein Vererbung der Klassen Screen und ViewModelBase, welche für das Marking Menu sowie für die Kommunikation zwischen View und ViewModel zuständig sind. In der Abbildung 13.15 wird die Struktur nochmals in einem UML-Diagramm verdeutlicht. Dabei ist zu erkennen, dass ViewModel-Objekte von ViewModelBase und Screen erben und es eine eins zu eins Beziehung zwischen View und ViewModel gibt. Die abschließende Auflistung beschreibt die jeweiligen Parameter nochmal detailiert.

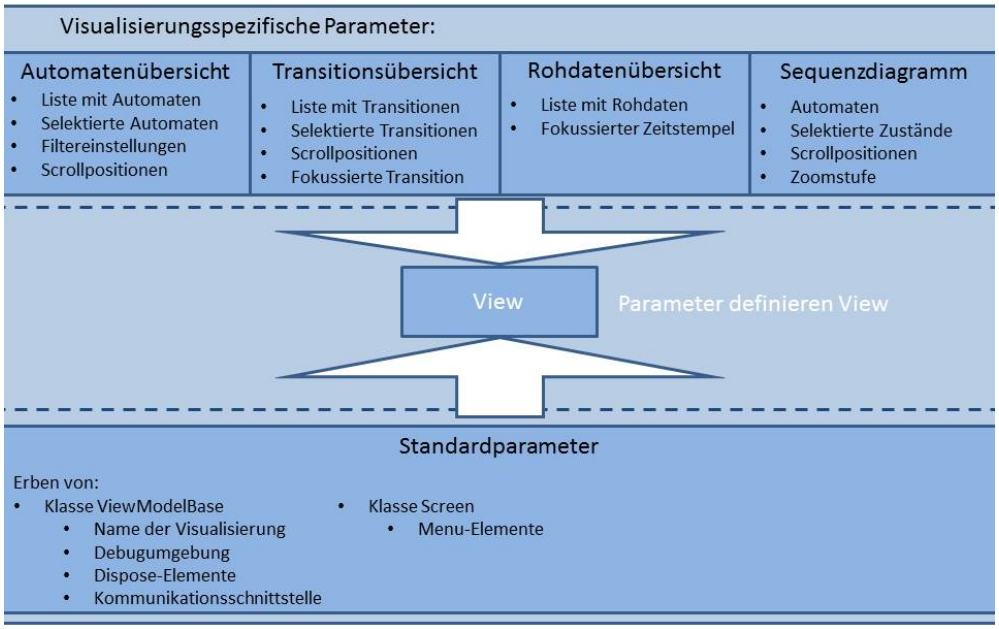


Abbildung 13.14: KnoVA Visualisierungsmodell

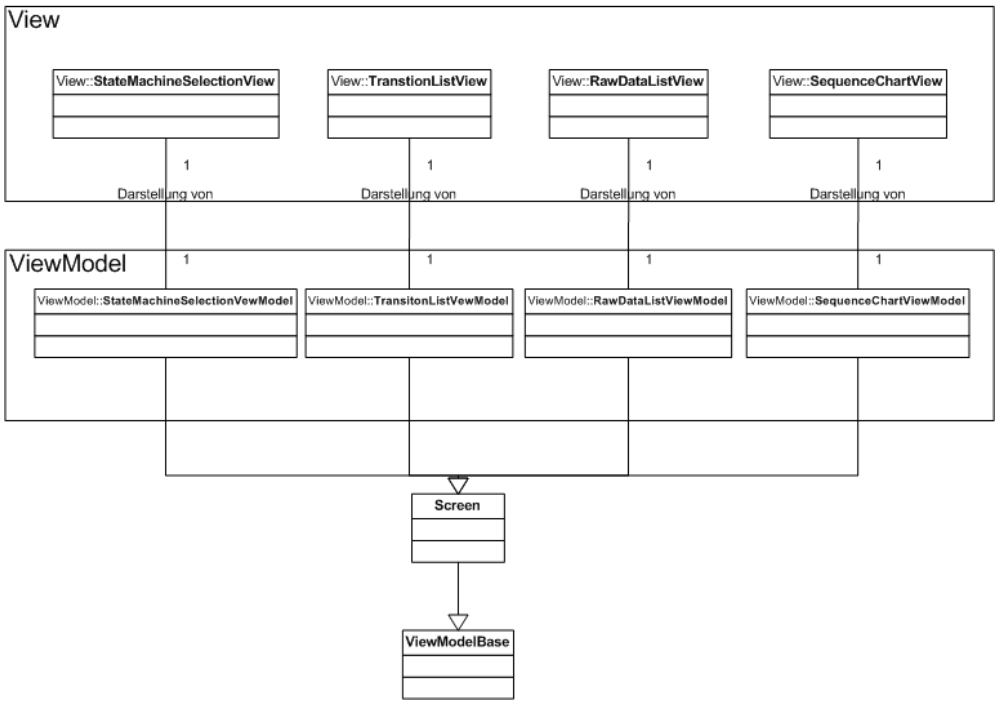


Abbildung 13.15: UML-Diagramm KnoVA Referenzmodell

- Strukturen der Klasse ViewModelBase:
  Es werden alle Strukturen der Klasse ViewModelBase geerbt, welche folgende Funktionen bietet:
  - Namen der Visualisierung
  - Debugumgebung
  - *Dispose*-Elemente für Visualisierungsobjekte
  - Kommunikationsschnittstelle zwischen View u. ViewModel über Events
- Strukturen der Klasse Screen:
  Über die Strukturen der Klasse Screen können Menü-Elemente zur Verfügung gestellt werden.

Automatenansicht Die Automatenansicht verfügt gegenüber den anderen Visualisierungen über folgende spezifische Parameter, um einen Ursprungszustand bzw. einen Wissenszeitpunkt wiederherzustellen.

- Liste mit allen Automaten die angezeigt werden. Diese lässt sich aus dem Testrun-Objekt ableiten.
- Menge an selektierten Automaten.
- Filtereinstellungen des gespeicherten Zustandes.
- Scrollpositionen auf der X- und Y-Achse.

Zustandssequenzdiagramm Das Zustandssequenzdiagramm verfügt gegenüber den anderen Visualisierungen über folgende spezifische Parameter, um einen Ursprungszustand bzw. einen Wissenszeitpunkt wiederherzustellen.

- Automat welcher das Zustandssequenzdiagramm anzeigen soll.
- Menge an selektierten Zuständen des Automaten.
- Scrollpositionen auf der Y-Achse.
- Zoomstufe im Zustandssequenzdiagramm.

Transitionsliste Die Transitionsliste verfügt gegenüber den anderen Visualisierungen über folgende spezifische Paramter, um einen Ursprungszustand bzw. einen Wissenszeitpunkt wiederherzustellen.

- Liste aller Transitionen die angezeigt werden. Diese stammt direkt aus dem Testrun-Objekt.
- Menge an selektierten Transitionen.

- Scrollpositionen auf der X- und Y-Achse.
- Fokussierte Transition in der Liste.

Rohdatenansicht Die Rohdatenansicht verfügt gegenüber den anderen Visualisierungen über folgende spezifische Parameter, um einen Ursprungszustand bzw. einen Wissenszeitpunkt wiederherzustellen.

- Liste mit allen Rohdaten die angezeigt werden. Diese stammt direkt aus dem Testrun-Objekt.
- Fokussierter Zeitstempel in den Rohdaten

13.4.2 Umsetzung im Prototypen

Die Umsetzung für die Wissensextraktion kann in drei Kategorien unterteilt werden. Diese beschreiben jeweils wie das Wissen übertragen bzw. abgelegt wird. In den einzelnen Kategorien finden sich Module des Prototypen wieder, welche die jeweiligen Funktionalitäten bieten.

Direktes Übertragen von Wissen

Das direkte Übertragen von Wissen ist speziell für die kollaborative Arbeit an einem Analysetisch interessant. Durch die Funktionen besteht die Möglichkeit Analysewissen von einem Spezialisten zum anderen zu übertragen. Anschließend wird dafür die Umsetzung vorgestellt.

Workbenchhandling Der Analyst hat die Möglichkeit durch Drehen oder Verschieben seines Analyseablaufes das Wissen mit einem anderen Analysten am Tisch zu teilen. Durch die Funktionen des Workbench-Handling ist dies möglich, da jedes Objekt des Analyseablaufes in einer frei bewegbaren Workbench ist. Diese Möglichkeit stellt allerdings keinen direkten Bezug auf das bereits vorgestellte Modell dar. Es werden keine Daten kopiert sondern nur verschoben, somit hat der Analyst auch nicht mehr seinen ursprünglichen Analyseablauf.

Synchronisierung Die Synchronisierung bietet für das direkte Übertragen von Wissen zwei Möglichkeiten. Das Spiegeln sowie das Kopieren(optional mit Pfad). Die genaue Umsetzung, im technischen Kontext wird in Abschnitt 13.3.3 erläutert. Im Bezug auf das vorgestellte Modell kann jedoch gesagt werden, dass die Synchronisierung aufgrund der halb-standardisierten Visualisierungen stark vereinfacht wurde.

Permanentes Ablegen von Wissen

Das permanente Ablegen von Wissen stellt eine Möglichkeit dar einen Analyseablauf oder spezifisches Wissen zu speichern. Dazu kann zu einem späteren Zeitpunkt, unabhängig vom Programm, nochmals auf die Daten zugegriffen werden.

Sessionmanagement Das Sessionmanagement stellt die Möglichkeit dar gesamte Analyseabläufe zu speichern und wiederherzustellen. Die genaue technische Umsetzung dazu wird in Kapitel 13.3.5 erläutert. Im KnoVA Kontext nutzt das Sessionmanagement auch den standardisierten Aufbau der Views. Durch diesen Aufbau ist es möglich die Daten persistent zu speichern und zu einem späteren Zeitpunkt wieder verfügbar zu machen. Zudem bietet der teil standardisierte Aufbau und die wenigen zusätzlichen Parameter keinen Overhead, welcher mit gespeichert werden muss. Dies erleichtert die Speicherung und den Austausch von Session-Dateien.

Filterung Die Filterung, technisch vorgestellt in Kapitel 13.3.6, bietet dem Nutzer die Möglichkeit Analysewissen direkt persistent zu speichern und wieder anzuwenden. Beim speichern dieses Analysewissens wird kein vollständiges Abbild der Daten gespeichert, sondern nur die Filtereinstellungen. Somit ist es möglich Filter auf andere, ähnliche Daten anzuwenden. Dies ist auch begründet auf dem standardisierten Modell, da die Daten, welche in den Views dargestellt werden zu einen großen Teil gleichartige Parameter besitzen.

Ad hoc-Speicherung des Wissens im Analyseablauf

Die Ad hoc-Speicherung des Analysewissens wird über eine History-Funktion zur Verfügung gestellt. So kann der Analyst seinen bisherigen Analyseverlauf immer nachvollziehen und durch das navigieren in der History rückgängig machen oder wiederherstellen.

History Die History bietet dem Analysten die Möglichkeit sein bisher angewendetes Wissen wieder verfügbar zu machen. Die technische Umsetzung wurde bereits in Kapitel 13.3.4 beschrieben. Die History nutzt dabei, wie das Sessionmanagement, den, über das Referenzmodell normierten, Aufbau der Views. Somit ist es möglich relativ wenig Daten mit zu sichern um einen Analyseablauf wieder vollständig herzustellen. Zudem können die gespeicherten Daten für das Sessionmanagement genutzt werden. Das Speicherintervall wurde für den Prototypen auf Änderungen von Daten begrenzt, da ansonsten eine Festlegung der

Granularität erfolgen müsste. Diese Daten manipulierenden Änderungen werden schrittweise nach dem vorgestellten Modell gespeichert.

Teil IV

Abschluss

Kapitel 14

Zusammenfassung und Ausblick

Mit TOAD wurde im Rahmen der Projektgruppe ein System zur kollaborativen visuellen Analyse von Fahrzeugbusdaten erstellt.

Das System kann von mehreren Analysten gleichzeitig über einen Multitouch-Tisch bedient werden. Es bietet unterschiedliche interaktive Visualisierungen der Fahrzeugbusdaten auf mehreren Abstraktionsniveaus. Diese Visualisierungen werden in Arbeitsbereichen, den so genannten Workbenches, dargestellt. Die Workbenches können dabei frei platziert, skaliert und rotiert werden, so dass der Tisch von allen Seiten bedienbar ist. Mehrere Nutzer können eine Analyse in diversen Workbenches durchführen. Diese so genannten Analysepfade können wiederum verschiedene Visualisierungen beinhalten, was durch eine Verknüpfung der Workbenches realisiert ist. Über eine Synchronisierungsfunktion ist es möglich einzelne Workbenches inklusive Interaktionen in andere Workbenches zu spiegeln oder ganze Pfade von verknüpften Workbenches zu kopieren. Mittels einer History kann über die visuelle Darstellung voriger Analysezustände zu diesen zurückgekehrt werden, wodurch einzelne Analyseschritte rückgängig gemacht werden. Eine Filterfunktion ermöglicht die Einschränkung der gesamten Datenmenge auf eine, durch Filter spezifizierte, Untermenge. Die Interaktion mit dem System findet über Gesten, also Bewegungen der Finger auf der Oberfläche des Multitouch-Tisches, statt.

Die Anforderungen des Systems wurden anhand tatsächlicher Analyseabläufe bei BMW ermittelt. Die enge Zusammenarbeit mit den Experten von BMW, und damit den eigentlichen Nutzern des Systems, ermöglichte die Entwicklung eines Systems, dass den Analyseprozess von Fahrzeugbusdaten möglichst optimal unterstützt. Dazu wurde das System in mehreren Iterationen des User Centered Design Prozesses, konkret des SCiVA Designprozesses, angepasst.

Neben den Evaluationen, die parallel zur Entwicklung des Systems durchgeführt wurden, erfolgte zum Ende des Projekts eine Vorstellung des Systems bei BMW. Das dabei gesammelte qualitative Feedback ergab, dass das von uns entwickelte System nutzbringende, bzw. effizienzsteigernde Visualisierungen und Interaktionskonzepte zur Unterstützung der kollaborativen visuellen Analyse von Fahrzeugbusdaten bietet.

Dennoch fanden sich sowohl in der Projektdurchführung als auch bei der Präsentation des Prototypen einzelne Punkte, an denen das System verbessert oder erweitert werden könnte. Hierauf wird im folgenden Ausblick näher eingegangen.

Das System bietet einen Ansatz zur kollaborativen visuellen Analyse von Fahrzeugbusdaten. Im Laufe des Projekts, besonders in den Evaluationen und Gesprächen mit BMW, konnten weitere Ideen und Ansatzpunkte zur Erweiterung und Verbesserung des Systems ermittelt werden.

Erweiterung des Bedienkonzepts: In den Evaluationen konnten zahlreiche Ideen zum Bedienkonzept gesammelt werden. Genannt sei hier beispielsweise die Frage ob beim Verschieben einer Workbench die verbundenen Workbenches, ggf. unter Berücksichtigung der Physik, mitverschoben werden sollen. Auch weitere Bedienkonzepte wie das *Verschmelzen* zweier Workbenches zum Vergleichen der enthaltenen Daten sind denkbar.

Nutzung einer Datenbank: Ursprünglich wurde geplant die zu analysierenden Rohdaten und weitere Informationen, wie zum Beispiel gespeicherte Analysesessions in einer SQL-Datenbank zu verwalten. Diese könnte dann auch die Funktionen eines Data Warehouse übernehmen. Es ist denkbar, bestimmte Ausprägungen der zu analysierenden Daten über mehrere Analyseprozesse zu aggregieren und damit eine Metaanalyse des zu Grunde liegenden Entwicklungsprozesses durchzuführen. Weiter wäre es denkbar, den Analyseprozess an sich automatisiert zu untersuchen um aus dem impliziten Wissen der Experten explizites Wissen zu generieren, das automatisch auf die zu analysierenden Daten angewendet werden könnte.

Remote-Kollaboration: Eine der größten zukünftigen Herausforderungen ist es, räumlich verteilten Teams die kollaborative Analyse zu ermöglichen. Teammitglieder mit verschiedenen Aufgaben sitzen häufig in unterschiedlichen Abteilungen und damit auch räumlich getrennt voneinander. Es ist vom Einsatzzweck und vom Aufbau des Systems her grundsätzlich denkbar, mehrere Endgeräte per Netzwerk/Internet miteinander zu verbinden, so dass auf diesen kollaborativ an einem Analyseprozess gearbeitet werden kann. So könnten räumliche Beschränkungen aufgehoben werden und Experten eines zu analysierenden Fachproblems über eine beliebige Distanz zusammen ein Problem bearbeiten. Da mit BMW in Zukunft eine weitere Forschungszusammenarbeit möglich ist, wäre es denkbar diese Themengebiete im Rahmen folgender Projektgruppen zu realisieren.

Kapitel 15

Statement PG

Gegen Ende des Wintersemesters 2009/2010 fiel unsere Wahl auf die PG VASC, da wir uns dadurch erhofften, Neues im Bereich der visuellen Analyse zu lernen und unsere Programmierkenntnisse auf gestengesteuerte Anwendungen auszuweiten. Die Kooperation mit BMW stellte einen weiteren Anreiz dar. Kurz nach Beginn der PG wurde zudem klar, dass wir darüber hinaus teilweise mit C# eine neue Programmiersprache erlernen würde.

Diese Hoffnungen wurden vollkommen erfüllt. Neben der Ausweitungen unserer fachlichen Kompetenzen als (Wirtschafts-)Informatikstudenten durften wir einige sehr sympathische und kompetente Personen kennenlernen. Außerdem waren die Einblicke in die Entwicklungsabläufe bei BMW im höchsten Maße interessant.

Die Zusammenarbeit mit BMW ermöglichten uns zudem Einblicke darin, wie ein Kooperationsprojekt mit einem Dritten ablaufen kann, und welche Dinge es dabei zu beachten gilt, insbesondere dann, wenn durch die räumliche Distanz direkte Treffen nur schwer oder gar nicht möglich sind.

Das Arbeits- und Sozialklima innerhalb der PG war immer sehr angenehm und Meinungsverschiedenheiten wurden auf konstruktive Weise aus der Welt geschafft. Alle Teilnehmer - und Betreuer - halfen sich im Verlauf gegenseitig, so dass stets Synergieeffekte stattfanden. Diese Effekte bezogen sich darauf, dass jeder von uns verschiedene Kompetenzen und Schwachpunkte hatte, die wir durch gegenseitiges Helfen ausglichen. Neben der eigentlichen Arbeit fanden immer wieder auch soziale Treffen statt, in denen wir unser Zusammengehörigkeitsgefühl festigen konnten.

Zusammenfassend kann man sagen, dass wir unsere fachlichen Kompetenzen erfolgreich erweitert haben und ein paar gute Leute kennengelernt haben, mit denen man immer wieder gerne Projekte durchführen würde.

Teil V

Anhang

Anhang A

Anwendungsfälle

Ein Anwendungsfall beschreibt, mit welchen Aktionen Analysten ein vorgegebenes Ziel erreichen können. Im Folgenden werden die Anwendungsfälle der benutzten Software beschrieben.

[UC.1] Ausführung einer *Tap*-Geste

Beschreibung	Der Akteur führt eine *Tap*-Geste aus.
Akteure	Analyst
Nachbedingungen	Der Akteur hat eine *Tap*-Geste durchgeführt.
Standardablauf	1. Der Akteur berührt kurz mit einem Finger einen Bereich (Beispielsweise einen Button).

[UC.2] Ausführung einer *Pinch*-Geste

Beschreibung	Der Akteur führt eine *Pinch*-Geste in einem bestimmten Bereich des Multitouch-Tisches aus.
Akteure	Analyst
Beinhaltet	*Move*-Geste
Nachbedingungen	Der Akteur hat eine *Pinch*-Geste durchgeführt.
Standardablauf	1. Der Akteur berührt mit mindestens zwei Fingern einen festgelegten Bereich. 2. Er verringert die Distanz zwischen beiden Fingern.

[UC.3] Ausführung einer *Spread*-Geste

Beschreibung	Der Akteur führt eine *Spread*-Geste in einem bestimmten Bereich des Multitouch-Tisches aus.
Akteure	Analyst
Beinhaltet	*Move*-Geste
Nachbedingungen	Der Akteur hat eine *Spread*-Geste durchgeführt.
Standardablauf	1. Der Akteur berührt mit mindestens zwei Fingern einen festgelegten Bereich. 2. Er vergrößert die Distanz zwischen beiden Fingern.

[UC.4] Ausführung einer *Rotate*-Geste

Beschreibung	Der Akteur führt eine *Rotate*-Geste in einem bestimmten Bereich des Multitouch-Tisches aus.
Akteure	Analyst
Beinhaltet	*Move*-Geste
Nachbedingungen	Der Akteur hat eine *Rotate*-Geste durchgeführt.
Standardablauf	1. Der Akteur berührt mit mindestens zwei Fingern einen festgelegten Bereich. 2. Er dreht alle Finger in eine Richtung relativ zum Mittelpunkt aller Finger.

[UC.5] Ausführung einer *Drag and Drop*-Geste

Beschreibung	Der Akteur führt eine *Drag and Drop*-Geste in einem bestimmten Bereich des Multitouch-Tisches aus.
Akteure	Analyst
Beinhaltet	*Move*-Geste
Nachbedingungen	Der Akteur hat eine *Drag and Drop*-Geste durchgeführt.

Standardablauf	1. Der Akteur berührt und hält mit mindestens einem Finger ein Objekt, welches für eine *Drag and Drop*-Geste benutzt werden kann. 2. Er bewegt den/die Finger auf dem Tisch zu dem Bereich, wo das Objekt abgelegt werden soll. 3. Bei Erreichen des Zielbereiches hebt er die Finger vom Tisch.

[UC.6] Ausführung einer *Press and Hold*-Geste

Beschreibung	Der Akteur führt eine Press and Hold-Geste in einem bestimmten Bereich des Multitouch-Tisches aus.
Akteure	Analyst
Beinhaltet	Tap-Geste
Nachbedingungen	Der Akteur hat eine Press and Hold-Geste durchgeführt.
Standardablauf	1. Der Akteur berührt mit einem Finger einen festgelegten Bereich und bewegt den Finger nicht.

[UC.7] Verschieben in der Visualisierung

Beschreibung	Der Akteur möchte in einer Visualisierung den betrachteten Datenausschnitt verschieben.
Akteure	Analyst
Beinhaltet	*Move*-Geste
Vorbedingungen	Workbench mit Visualisierung geöffnet
Nachbedingungen	Der betrachtete Datenausschnitt hat sich verändert.
Standardablauf	1. Der Akteur berührt mit mindestens einem Finger die Visualisierung 2. Er führt eine *Move*-Geste aus. 3. Die Visualisierung verschiebt sich während der Bewegung in die Richtung der Geste.

[UC.8] Hinein-Zoomen in der Visualisierung

Beschreibung	Der Akteur möchte einen Ausschnitt der Visualisierung detaillierter betrachten. Hierzu führt er eine *Spread*-Geste auf der Visualisierung durch.
Akteure	Analyst
Beinhaltet	*Spread*-Geste
Vorbedingungen	Eine Visualisierung wurde geöffnet.
Nachbedingungen	Es wird ein detaillierterer Ausschnitt der visualisierten Daten angezeigt.
Standardablauf	1. Der Akteur führt eine *Spread*-Geste auf der Visualisierung aus. 2. Das Programm skaliert die Visualisierung hoch, ohne die Maße der Workbench zu verändern. Je nach Zoom-Level wird eine Visualisierung geladen, die den Datenausschnitt detaillierter darstellen kann, beispielsweise werden zusammengefasste Daten in einer detaillierteren Ansicht nicht mehr zusammengefasst.

[UC.9] Hinaus-Zoomen in der Visualisierung

Beschreibung	Der Akteur möchte einen gröberen Ausschnitt der visualisierten Daten betrachten. Hierfür führt er eine *Pinch*-Geste auf der Oberfläche des Diagramms durch.
Akteure	Analyst
Beinhaltet	*Pinch*-Geste
Vorbedingungen	Eine Visualisierung wurde geöffnet.
Nachbedingungen	Es wird ein gröberer Ausschnitt der visualisierten Daten angezeigt.
Standardablauf	1. Der Akteur führt eine *Pinch*-Geste auf der Visualisierung aus. 2. Das Programm skaliert die Visualisierung herunter, ohne die Maße der Workbench zu verändern. Je nach Zoom-Level wird eine Visualisierung geladen, die den Datenausschnitt gröber darstellen kann, beispielsweise werden Daten in einer gröberen Ansicht zusammengefasst.

[UC.10] Verschieben der Workbench

Beschreibung	Der Akteur möchte eine Workench auf dem Tisch in eine andere Richtung bewegen.
Akteure	Analyst

Beinhaltet	*Move*-Geste
Vorbedingungen	Workbench geöffnet
Nachbedingungen	Die Workbench ist an der Position, an der der Nutzer sie haben möchte.
Standardablauf	1. Der Akteur berührt mit mindestens einem Finger die Workbench außerhalb einer eventuell geladenen Visualisierung. 2. Er führt eine *Move*-Geste aus. 3. Die Worbench verschiebt sich während der Bewegung in die Richtung der Geste.
Alternativablauf	Im ersten Punkt kann der Bereich, der berührt werden soll auch auf die Ecken der Workbench beschränkt werden.

[UC.11] Auswahl einer Datenquelle aus einer Datenbank

Beschreibung	Der Analyst wählt eine Datenquelle für die Analyse aus einer Datenbank.
Akteure	Analyst
Beinhaltet	Öffnen Tap-Menü
Vorbedingungen	1. Eine Workbench ist geöffnet.
Nachbedingungen	Die Datenquelle für die Analyse wurde aus der Datenbank geladen.
Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur wählt den Punkt zur Auswahl einer Datenquelle im Tap-Menü. 3. In der nächsten Hierachiestufe wählt der Akteur den Punkt „Datenbank“. 4. Eine Liste der verfügbaren Datenbankverbindungen wird angezeigt. 5. Der Akteur wählt eine der angezeigten Verbindungen aus. 6. Der Akteur bestätigt seine Auswahl. 7. Die Datenquelle wird für die Analyse geladen.
Alternativablauf	Eine neu erstellte Workbench bietet direkt die Möglichkeit zur Auswahl einer Datenquelle. Hierfür muss das Tap-Menü nicht mehr geöffnet werden. Der Akteur wählt lediglich aus, dass die Datenquelle aus einer Datenbank geladen werden soll. Der weitere Ablauf ist mit dem Standardablauf identisch.

[UC.12] Auswahl einer Datenquelle aus einer XML-Datei

Beschreibung	Der Analyst wählt eine Datenquelle für die Analyse aus einer XML-Datei.
Akteure	Analyst
Beinhaltet	Öffnen Tap-Menü
Vorbedingungen	1. Eine Workbench ist geöffnet.
Nachbedingungen	1. Die Daten aus der XML-Datei wurden in die Datenbank geladen. 2. Die in die Datenbank geladenen Daten werden für die Analyse aus wieder aus der Datenbank geladen.
Ausnahmen	Die Daten in der XML-Datei können nicht in das Datenbankschema überführt und somit weder in die Datenbank noch für die Analyse geladen werden. Der Akteur bekommt in diesem Fall eine Fehlermeldung angezeigt.
Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur wählt den Punkt zur Auswahl einer Datenquelle im Tap-Menü. 3. In der nächsten Hierachiestufe wählt der Akteur den Punkt „Daten aus XML-Datei“. 4. Ein Dateiauswahldialog wird angezeigt. 5. Der Akteur wählt die zu ladende XML-Datei aus. 6. Der Akteur bestätigt seine Auswahl. 7. Eine Liste der verfügbaren Datenbankverbindungen wird angezeigt. 8. Der Akteur wählt eine Datenbankverbindung zu der Datenbank aus, in die die Daten aus der XML-Datei geladen werden soll. 9. Der Akteur bestätigt seine Auswahl 10. Die XML-Daten werden zunächst in die Datenbank geladen. Anschließend werden die Daten für die Analyse wieder aus der Datenbank geladen.
Alternativablauf	Eine neu erstellte Workbench bietet direkt die Möglichkeit zur Auswahl einer Datenquelle. Hierfür muss das Tap-Menü nicht mehr geöffnet werden. Der Akteur wählt lediglich aus, dass die Datenquelle aus einer XML-Datei geladen werden soll. Der weitere Ablauf ist mit dem Standardablauf identisch.

[UC.13] Speichern einer Session von verknüpften Workbenches

Beschreibung	Der Analyst speichert eine Session. Eine Session beinhaltet die jeweilige Workbench sowie alle mit dieser Workbench verknüpften Workbenches und die Verknüpfungen dazwischen.
Akteure	Analyst
Beinhaltet	Öffnen Tap-Menü
Vorbedingungen	1. Eine Workbench ist geöffnet.
Nachbedingungen	Eine Session wurde so gespeichert, dass sie später wiederhergestellt werden kann.
Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur wählt den Punkt zur Speicherung einer Session im Tap-Menü. 3. Ein Textfeld wird angezeigt. 4. Der Akteur gibt in dem Textfeld den Namen der zu speichernden Session an. 5. Der Akteur bestätigt seine Eingabe. 6. Die Session wird gespeichert.

[UC.14] Laden einer Session von verknüpften Workbenches

Beschreibung	Der Analyst lädt eine Session. Eine Session beinhaltet die jeweilige Workbench sowie alle mit dieser Workbench verknüpften Workbenches und die Verknüpfungen dazwischen.
Akteure	Analyst
Beinhaltet	Öffnen Tap-Menü
Vorbedingungen	1. Eine Workbench ist geöffnet.
Nachbedingungen	Eine ausgewählte Session wurde geladen. Die darin enthaltenen Workbenches und Verknüpfungen werden wie zum Speicherzeitpunkt dargestellt.

Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur wählt den Punkt zum Laden einer Session im Tap-Menü. 3. Eine Liste der verfügbaren gespeicherten Sessions wird angezeigt. 4. Der Akteur wählt eine der angezeigten Sessions aus. 5. Der Akteur bestätigt seine Auswahl. 6. Die Session wird geladen.

[UC.15] Einblenden der History

Beschreibung	Der Analyst blendet die ausgeblendete History einer Workbench wieder ein.
Akteure	Analyst
Vorbedingungen	1. Eine Workbench ist geöffnet. 2. Die History der Workbench ist ausgeblendet.
Nachbedingungen	Die History der Workbench ist sichtbar.
Standardablauf	1. Der Akteur berührt die Schaltfläche zum Ein-/Aublenden der History innerhalb einer Workbench. 2. Die History der Workbench wird eingeblendet.

[UC.16] Ausblenden der History

Beschreibung	Der Analyst blendet die History einer Workbench aus.
Akteure	Analyst
Vorbedingungen	1. Eine Workbench ist geöffnet. 2. Die History der Workbench ist sichtbar.
Nachbedingungen	Die History der Workbench ist ausgeblendet.

Standardablauf	1. Der Akteur berührt die Schaltfläche zum Ein-/Aublenden der History innerhalb einer Workbench. 2. Die History der Workbench wird ausgeblendet, das heißt sie ist nicht mehr sichtbar.

[UC.17] Export Ergbenisse

Beschreibung	Der Anwender kann jederzeit seine aktuellen Analyseergebnisse exportieren, um sie z.B. zu präsentieren.
Akteure	Analyst
Beinhaltet	Öffne Tap-Manü
Vorbedingungen	1. Eine Datenbasis wurde gewählt.
Nachbedingungen	Der Akteur hat eine Ansicht zum Export bestimmt.
Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur wählt die Option „Exportieren“. 3. Der Akteur bestimmt einen Dateinamen und das Dateiformat. 4. Der Akteur bestätigt seine Eingaben mit Druck auf die Schaltfläche „OK“.

[UC.18] Anzeige der Rohdaten

Beschreibung	Der Anwender kann jederzeit zu den ausgewählten Automaten und/oder Transitionen die Rohdaten aufrufen
Akteure	Analyst
Beinhaltet	Öffne Tap-Menü
Vorbedingungen	1. Eine Datenbasis wurde gewählt.
Nachbedingungen	Der Akteur erhält die Ansicht der Rohdaten.
Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur wählt die Option „Rohdaten“.

[UC.19] Sortierung der Listen-Ansicht

Beschreibung	Der Anwender kann die Listen-Ansicht nach den Kriterien „Zeitstempel“, „Zustandsart“, „Startzustand“ und „Endzustand“ sortieren.
Akteure	Analyst
Vorbedingungen	1. Eine Datenbasis wurde gewählt. 2. Eine Listen-Ansicht wird angezeigt.
Nachbedingungen	Die Listen-Ansicht ist neu sortiert.
Standardablauf	1. Der Akteur wählt eines der Kriterien, nach denen er sortieren möchte. 2. Der Akteur tippt auf das entsprechende Kriterium.

[UC.20] Öffnen des TAP-Menüs

Beschreibung	Das TAP-Menü dient im Allgemeinen zur Navigation im Programm. Der Inhalt des TAP-Menüs ist abhängig von der aktuellen Visualisierung in der das TAP-Menü aufgerufen wird.
Akteure	Analyst
Beinhaltet	Press and Hold-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Das TAP-Menü ist noch nicht angezeigt.
Standardablauf	1. Das Akteur führt eine Press and Hold-Geste auf einer Workbench aus. 2. Das TAP-Menü öffnet sich. 3. Der Akteur kann im Menü die gewünschte Option auswählen.

[UC.21] Wechsel der Darstellungsart in der Automatenauswahl

Beschreibung	In der Automatenauswahlvisualisierung existieren verschiedene Arten zur Darstellung der einzelnen Automaten. Die Zustände innerhalb eines Automaten können einerseits in Reihenfolge ihres Auftretens angezeigt werden und andererseits ist es möglich, die Zustände in Reihenfolge ihrer Relevanz für den Analysten anzuzeigen (Info → Okay → Warning → Defect). Weiterhin ist es möglich alle Automaten mit gleicher Breite untereinander anzeigen zu lassen und es ist auch möglich, die Automaten in Abhängigkeit zu einer Zeitachse anzuordnen.
Akteure	Analyst
Beinhaltet	Tap-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Die aktuelle Visualisierung innerhalb der Workbench ist die Automatenauswahl.
Standardablauf	1. Der Akteuer drückt den Button zum Wechsel der Darstellungsart der Automaten. 2. Der Akteuer wählt die gewünschte Darstellungsart aus.

[UC.22] Wechsel der Visualisierungsart im Zustands-Sequenzdiagramm

Beschreibung	Beim Zustands-Sequenzdiagramm kann der Akteur die Visualisierungsart zwischen horizontaler und vertikaler Ausrichtung wechseln.
Akteure	Analyst
Beinhaltet	Rotate-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Die aktuelle Visualisierung ist das Zustands-Sequenzdiagramm.
Standardablauf	1. Der Akteur führt eine Rotate-Geste innerhalb des Zustands-Sequenzdiagramm aus. 2. Die Visualisierung wechselt zwischen horizontaler und vertikaler Ausrichtung.

[UC.23] Point of Interest setzen

Beschreibung	Der Akteur kann bei jeder Visualisierung über das TAP-Menü einen Point of Interest setzen. Ein Point of Interest beinhaltet hierbei eine farbige Markierung einen Textkommentar.
Akteure	Analyst
Beinhaltet	Öffnen des TAP-Menüs
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt.
Standardablauf	1. Der Akteur öffnet das TAP-Menü. 2. Der Akteur wählt im TAP-Menü die Option „Point of Interest setzen“ aus.

[UC.24] Point of Interest löschen

Beschreibung	Der Akteur kann bei jeder Visualisierung die bestehenden Points of Interest löschen. Das Löschen ist möglichen indem der Akteur eine Press and Hold-Geste auf dem zu löschenden Point of Interest durchführung, somit das TAP-Menü öffnet und in diesem Menü dann die Option „Point of Interest löschen“ anwählt.
Akteure	Analyst
Beinhaltet	Öffnen des TAP-Menüs
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Ein Point of Interest wurde gesetzt.
Standardablauf	1. Der Akteur öffnet das TAP-Menü auf dem zu löschenden Point of Interest. 2. Der Akteur wählt im TAP-Menü die Option „Point of Interest löschen“ aus.

[UC.25] Verknüpfung Workbench erstellen

Beschreibung	Der Akteur kann manuell zwei bis n Workbenches verknüpfen um eine Synchronisation zwischen diesen auf verschiedenen Ebenen zu erreichen.
Akteure	Analyst
Vorbedingungen	Zwei oder mehr Workbeches sind vorhanden und können verknüpft werden
Nachbedingungen	Die für die Verknüpfung ausgewählten Workbenches sind auf der gewählten Ebene mit den vorgenommenen Einstellungen verbunden
Ausnahmen	Falls die Workbenches auf einer nicht verträglichen Verknüpfungsebene stattfinden findet keine Verknüpfung statt
Standardablauf	1. Der Akteur berührt die zu verknüpfende Workbench und zieht eine Linie zu der Workbench mit der synchronisiert werden soll 2. In einer Menüauswahl wählt der Akteur die Synchronisierungsen aus. 3. Die Synchronisierung mit den ausgewählten Einstellungen wird vom System vorgenommen

[UC.26] Verknüpfung zwischen Workbenches auflösen

Beschreibung	Der Akteur kann manuell die Verknüpfung zwischen Workbenches auflösen um eine Synchronisation zwischen diesen aufzulösen.
Akteure	Analyst
Vorbedingungen	Mindestens zwei Workbeches sind vorhanden und verknüpft
Nachbedingungen	Die Verknüpfung zwischen den gewählten Workbenches wurde auf den gewählten Ebenen aufgelöst
Ausnahmen	Falls keine Verknüpfung von der gewählten Workbench ausgehend existiert wird daStsynchronisierungsmenü nicht angezeigt
Standardablauf	1. Der Akteur berührt die Workbench deren Verknüpfung gelöst werden soll 2. In einer Menüauswahl wählt der Akteur die Synchronisierungsebenen aus zwischen denen die Verknüpfung gelöst werden soll. 3. Die Synchronisierung mit den ausgewählten Einstellungen wird vom System gelöst

[UC.27] Anzeige Listen-Ansicht bei Auswahl von Transition

Beschreibung	Der Akteur kann manuell in dem er innerhalb der Zustands-Sequenzdiagramm-View einen Finger auf einen Zustandsübergang legt die Listen-Ansicht einblenden lassen in der die Reihe der gewählten Transition markiert ist.
Akteure	Analyst
Vorbedingungen	Zustands-Sequenzdiagramm-View ist geöffnet und ein Finger liegt auf einer Transition
Nachbedingungen	Die Listen-Ansicht mit markierter Zeile wird angezeigt
Ausnahmen	
Standardablauf	1. Der Akteur berührt einen Transisitonsübergang innerhalb der Zustands-Sequenzdiagramm-View 2. Die Listen-Ansicht mit markierter Zeile des gewählten Zustandsübergangs wird eingeblendet 3. Die Listen-Ansicht wird vom Akteur wieder geschlossen

[UC.28] Automatenreihenfolge ändern

Beschreibung	Der Akteur kann manuell in der Automatenübersicht die Anzeigereihenfolge der Automaten ändern in dem er die Automaten per Drag and Drop an die neue Anzeigestelle verschiebe. Diese kann zwischen zwei angezeigten Automaten sein oder am Anfang oder Ende der jeweiligen Automatenübersichtsspalte.
Akteure	Analyst
Vorbedingungen	Automatenübersicht ist geöffnet und es befindet sich mehr als ein Automat in der Übersicht
Nachbedingungen	Die Automatenübersicht mit der neuen Automatenreihenfolge wird angezeigt
Ausnahmen	
Standardablauf	1. Der Akteur berührt einen Automaten innerhalb der Automatenübersicht 2. Der Akteur verschiebt den gewählten Automaten an eine neue Position innerhalb der Automatenübersicht 3. Die neue Anzeigereihenfolge wird vom System angezeigt

[UC.29] Vergrößern der Workbench

Beschreibung	Diese Funktion dient zum Vergrößern einer Workbench
Akteure	Analyst
Beinhaltet	*Spread*-Geste
Vorbedingungen	Eine Workbench wurde bereits gestartet.
Nachbedingungen	Das Diagramm wurde vergrößert.
Standardablauf	1. Der Akteur berührt und hält einen Eckpunkt einer Workbench mit einem Finger. 2. Er bewegt den Finger von der Workbench weg, bis die gewünschte Größe erreicht wurde. 3. Beim Erreichen der gewünschten Größe hebt er den Finger.

[UC.30] Verkleinern der Workbench

Beschreibung	Diese Funktion dient zum Verkleinern einer Workbench
Akteure	Analyst
Beinhaltet	*Pinch*-Geste
Vorbedingungen	Eine Workbench wurde bereits gestartet.
Nachbedingungen	Die Größe des Diagramms wurde verringert.
Standardablauf	1. Der Akteur berührt zwei gegenüberliegende Ecken des Diagramms. 2. Er führt eine *Pinch*-Geste aus. 3. Das Programm skaliert das Diagramm herunter. Die Inhalte des Diagramms passen sich prozentual an die Größe an.

[UC.31] Workbench erstellen

Beschreibung	Der Benutzer erstellt nach dem Start des Programms eine neue Workbench.
Akteure	Analyst
Beinhaltet	*Move*-Geste
Vorbedingungen	Das Programm ist gestartet.
Nachbedingungen	Eine neue Workbench wurde erstellt.

Standardablauf	1. Der Analyst startet die Anwendung. 2. Er führt eine *Move*-Geste zu sich aus. Das bedeutet er zieht mit einem Finger eine Linie auf der freien Fläche. 3. Die Workbench wird in der Ausrichtung zum Analysten und mit der Größe der Linie erstellt.

[UC.32] Workbench schließen

Beschreibung	Der Benutzer schließt eine existierende Workbench.
Akteure	Analyst
Beinhaltet	*Tap*-Geste
Vorbedingungen	Eine Workbench wurde bereits gestartet.
Nachbedingungen	Die Workbench wurde geschlossen.
Standardablauf	1. Der Akteur bedient einen Button zum Schließen mit der *Tap*-Geste. 2. Die Workbench wird unwiderruflich geschlossen.

[UC.33] Workbench rotieren

Beschreibung	Diese Funktion dient zum Drehen einer Workbench
Akteure	Analyst
Beinhaltet	*Drag and Drop*-Geste, *Pinch*-Geste
Vorbedingungen	Eine Workbench wurde bereits gestartet.
Nachbedingungen	Der Akteur hat die Workbench gedreht.
Standardablauf	1. Der Akteur berührt zwei weit auseinander liegende Punkte auf der Workbench. 2. Er führt eine *Pinch*-Geste aus und rotiert die Workbench in die beliebige Position. 3. Bei Erreichen der gewünschten Drehung hebt er die Finger.

[UC.34] Navigieren in der History

Beschreibung	Der Akteur navigiert durch die History und kann verschiedene Zustände laden.
Akteure	Analyst
Beinhaltet	Move-Geste und Spread-/Pinch-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Der Akteur hat bereits Analyseschritte durchgeführt.
Standardablauf	1. Der Akteur öffnet die History. 2. Der Akteur navigiert mit Hilfe einer Move-Geste durch die History. 3. Der Akteur kann mit einer Spread-/Pinch-Geste den History-Bereich verkleinern und vergrößern.

[UC.35] Laden eines Zustandes aus History

Beschreibung	Ein Zustand wird aus der History geladen und auf der Workbench ausgeführt.
Akteure	Analyst
Beinhaltet	Tap-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Der Akteur hat bereits Analyseschritte durchgeführt. 4. Der Akteur hat die History geöffnet.
Standardablauf	1. Der Akteur navigiert durch die History. 2. Der Akteur Wählt einen Zustand mit Hilfe einer Tap-Geste aus. 3. Der Akteur bestätigt die Auswahl.

[UC.36] Anwählen von Automaten

Beschreibung	Ein oder weitere Automaten werden dem Analyseprozess hinzugefügt.
Akteure	Analyst
Beinhaltet	Tap-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Die aktuelle Visualisierung ist die Automatenansicht.
Standardablauf	1. Der Akteur markiert mit Hilfe einer Tap-Geste die Automaten die dargestellt/ausgewählt werden sollen.

[UC.37] Abwählen von Automaten

Beschreibung	Bereits ausgewählte Automaten werden wieder entfernt.
Akteure	Analyst
Beinhaltet	Tap-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Die aktuelle Visualisierung ist die Automatenansicht.
Standardablauf	1. Der Akteur demarkiert mit Hilfe einer Tap-Geste die Automaten, welche für den Analyseprozess nicht weiter benötigt werden.

[UC.38] Wechsel der Visualisierungsansicht durch Tap-Menü

Beschreibung	Der Akteur wechselt die Visualisierungsansicht mit Hilfe des Tap-Menüs. Beispielsweise kann der Akteur vom Zustandssequenzdiagramm in die Listenansicht wechseln.
Akteure	Analyst
Beinhaltet	Öffnen des Tap-Menüs und Tap-Geste

Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt.
Standardablauf	1. Der Akteur öffnet das Tap-Menü. 2. Der Akteur durchläuft die Schritte im Tap-Menü. 3. Der Akteur wählt mit Hilfe einer Tap-Geste die Visualisierung aus, auf welche gewechselt werden soll.

[UC.39] Wechsel der Visualisierungsansicht durch Zooming

Beschreibung	Der Akteur wechselt die Visualisierungsansicht von der Automatenauswahl zum Zustandssequenzdiagramm, indem er Geste auf einem ausgewählten Automaten durchführt.
Akteure	Analyst
Beinhaltet	Spread-Geste
Vorbedingungen	1. Das Programm wurde gestartet. 2. Eine Workbench wurde erstellt. 3. Die aktuelle Visualisierung ist die Automatenauswahl. 4. Es sind bereits Automaten zur Visualisierung markiert.
Standardablauf	1. Der Akteur führt die Spread-Geste auf einen Automaten aus. 2. Durch den Zoom auf einen Automaten gelangt der Akteur in das Zustandssequenzdiagramm.

)

Literatur

[AG11a] AG, Doodle: *Doodle: easy scheduling*. http://www.doodle.com/. Version: 03 2011

[AG11b] AG, MEDION: *MEDION(R) The Touch x9613*. http://www.thetouch.de/medion/de/x9613/. Version: 03 2011

[al01] al, Beck K.: *Manifesto for Agile Software Development*. http://agilemanifesto.org/. Version: 2001

[Ang05] Angermeier, G.: *Projektmanagement Lexikon*. Projekt Magazin, 2005

[App11] Apple: *Apple - iPhone - iOS 4 ist das weltweit fortschrittlichste Betriebssystem fuer Mobilgeraete*. http://www.apple.com/de/iphone/ios4/. Version: 03 2011

[B.09] B., Gloger: *Scrum: Produkte zuverlässig und schnell entwickeln*. München, Hanser Verlag, 2009

[BA04] Beck, Kent ; Andres, Cynthia: *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004. – ISBN 0321278658

[BBVB+01a] Beck, K. ; Beedle, M. ; Van Bennekum, A. ; Cockburn, A. ; Cunnimgham, W. ; Fowler, M. ; Grenning, J. ; Highsmith, J. ; Hunt, A. ; Jeffries, R. u. a.: *Principles behind the agile manifesto*. http://www.agilemanifesto.org/principles.html. Version: 2001. – Abrufdatum: 21. März 2011

[BBVB+01b] Beck, K. ; Beedle, M. ; Van Bennekum, A. ; Cockburn, A. ; Cunningham, W. ; Fowler, M. ; Grenning, J. ; Highsmith, J. ; Hunt, A. ; Jeffries, R. u. a.: *Manifesto for agile software development*. http://www.agilemanifesto.org/. Version: 2001. – Abrufdatum: 21. März 2011

[BDR00] Buziek, G. ; Dransch, D. ; Rase, WD: *Dynamische Visualisierung*. Springer, 2000

[Bec99] Beck, Kent: *Extreme programming explained: embrace change*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999. – ISBN 0–201–61641–6

[BGM04] Bederson, Benjamin B. ; Grosjean, Jesse ; Meyer, Jon: Toolkit Design for Interactive Structured Graphics. In: *IEEE Trans. Softw. Eng.* 30 (2004), August, 535–546. http://dx.doi.org/10.1109/TSE.2004.44. – DOI 10.1109/TSE.2004.44. – ISSN 0098–5589

[BP06] Bach, M. ; Poloschek, CM.: Optical illusions. In: *ACNR* 6 (2006), S. 20–21

[Bro] Brooke, John: *SUS - A quick and dirty usability scale*. http://hell.meiert.org/core/pdf/sus.pdf

[Bux11] Buxton, Bill: *Multi-Touch Systems that I Have Known and Loved*. http://www.billbuxton.com/multitouchOverview.html. Version: 02 2011

[CFK+10] Cramer, Markus ; Fortmann, Jutta ; Klingenberg, Thole ; Maschke, Kai ; Puchkovskiy, Alexandr ; Rusinov, Plamen ; Schmitt, Marwin ; Sleumer, Konstantin ; Uphoff, Arne: *Abschlussbericht Projektgruppe Visual Analytics*. http://tap.informatik.uni-oldenburg.de/downloads/abschlussbericht.pdf. Version: 03 2010

[Der05] Dersteler, Juan C.: *Software Toolkits for Infovis*. http://www.infovis.net/printMag.php?num=162&lang=2. Version: 2005

[DFAB04] Dix, A. ; Finlay, J. ; Abowd, G.D. ; Beale, R.: *Human-Computer Interaction*. http://www.hcibook.com/e3/. Version: 2004

[DIN09] DIN: *DIN 69901-5:2009-01: Projektmanagement - Projektmanagementsysteme*. 2009

[e-t07] e-teaching.org: *Gestaltgesetze*. http://www.e-teaching.org/didaktik/gestaltung/visualisierung/gestaltgesetze/. Version: 2007. – [Online; Stand 8. Juni 2010]

[EK08] Eugen Kusmaul, Gewerbliche Schule Schwaebisch H.: *Touchscreen*. http://referate.mezdata.de/sj2007/10touchscreen_eugen-kusmaul/index.php. Version: 02 2008

[Ell03] Ellson, John: *Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools*. http://www.graphviz.org/Documentation/EGKNW03.pdf. Version: 2003

[Ell10] Eller, Frank: *Visual CSharp 2010: Grundlagen, Programmiertechniken, Datenbanken*. Addison-Wesley, 2010

[Fac11] Factolex.com: *Kapazitiver Sensor Fakten . Factolex - das Fakten Lexikon . erklaerungen kurz und praegnant*. http://de.factolex.com/Kapazitiver_Sensor. Version: 03 2011

[Fei03] Feitelson, D.: *Comparing partitions with spie charts, Essay*. University of Jerusalem, 2003

[FH01] Fowler, M. ; Highsmith, J.: The agile manifesto. In: *Software Development* 9 (2001), Nr. 8, S. 28–35. – ISSN 1070–8588

[FH10] Flöring, S. ; Hesselmann, T.: Tap: Towards visual analytics on interactive surfaces. In: *Collaborative Visualisations and interactive surfaces - CoVIS´09* (2010)

[FHJA11] Flöring, S. ; H.-Jürgen Appelrath, Prof. Dr. D.: KnoVA: introducing a reference model for knowledge-based visual analytics, 2011

[Fli07] Flickr: *WebTheme: ThemeView*. http://www.flickr.com/photos/pnnl/3655739040. Version: 2007. – [Online; Stand 4. Juni 2010]

[Fow01] Fowler, M.: The new methodology. In: *Wuhan University Journal of Natural Sciences* 6 (2001), Nr. 1, S. 12–24. – ISSN 1007–1202

[GK03] Gegenfurtner, K.R. ; Kiper, D.C.: Color Vision. In: *Annual review of neuroscience* 26 (2003), S. 181–206

[Gmb10a] GmbH, Statista: *Touch Screen Technologie - Marktanteile . Statistik*. http://de.statista.com/statistik/daten/studie/158326/umfrage/marktanteile-von-displays-nach-touch-screen-technologie/. Version: 05 2010

[Gmb10b] GmbH, Weka Media P.: *Smartphone-Betriebssysteme im Vergleich - connect - Magnus.de*. http://www.connect.de/ratgeber/smartphone-betriebssysteme-im-vergleich-1006883,945.html. Version: 08 2010

[Gus96] Guski, R.: Wahrnehmen - ein Lehrbuch. In: *Stuttgart: Kohlhammer* (1996). http://eco.psy.ruhr-uni-bochum.de/download/Guski-Lehrbuch/Inhaltsverzeichnis.html

[HBH10] Hesselmann, Tobias ; Boll, Susanne ; Heuten, Wilko: SCIVA - Designing Applications for Surface Computers. In: *In Proc. EICS 2011. ACM Press. To appear*. Pisa, Italy, 2010

[Hei10] Heinemann, Gerrit: *Der neue Online-Handel. Erfolgsfaktoren und Best Practices*. Gabler Verlag | Springer Fachmedien Wiesbaden GmbH, 2010

[HF64] Hommer, K. ; Frey, RG: Einzelreiz-und Flimmer-ERG im akuten Stadium der Chininvergiftung. In: *Documenta Ophthalmologica* 18 (1964), Nr. 1, S. 392–403

[HHE02] Highsmith, Jim ; Highsmith, James A. ; Eastlake, Donald: *Agile Software Development Ecosystems (Agile Software Development Series)*. Addison-Wesley Longman, Amsterdam, 2002. – ISBN 9780201760439

[HHI07] Helmke, Hartmut ; Höppner, Frank ; Isernhagen, Rolf: *Einführung in die Software-Entwicklung: Vom Programmieren zur erfolgreichen Software-Projektarbeit*. 1. Hanser Fachbuchverlag, 2007. – ISBN 9783446409699

[Hic09] Hicks, M.: Perceptual and Design Principles for Effective Interactive Visualisations. In: *Trends in Interactive Visualization* (2009), S. 155–174

[IGD07] IGD, Fraunhofer: *Die visuelle Analyse - Eine Einführung*. `http://www.igd.fraunhofer.de/igd-a3/extern/va-day/2007/pdfs/Kohlhammer.pdf`. Version: 2007. – [Online; Stand 3. Juni 2010]

[IIG07] Innovative Informatik GmbH oose: *Agiles Software-Projektmanagement, Trainingsunterlagen*. 2007

[Inf09] Informatik, Oose I.: Einfluss klassischer und agiler Techniken auf den Erfolg von IT-Projekten. (2009), 7. `http://www.oose.de/fileadmin/Dateien/Publikationen/Ergebnisbericht_Projektmanagementstudie.pdf`

[Inf10] Informatik, Oose I.: *Glossar*. `http://www.oose.de/oep/desc/glo-ov.htm`. Version: 2010

[Ins11] Institut, Fraunhofer: *Fraunhofer Institut - Visual Analytics*. `http://www.fraunhofer.de/forschungsthemen/fraunhofer-zukunftsthemen/visual-analytics.jsp`. Version: 03 2011

[Int11] Integration, OBJENTIS S.: *VISUELLE DATEN-EXPLORATION (VDX)*. `http://www.objentis.com/fileadmin/user_upload/Downloads/VDX/Visuelle-Daten-Exploration_Info-Text_V2_0.pdf`. Version: 03 2011

[JC05] J., Thomas ; Cook, K.: *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE, 2005

[JN10] Jakob Nielsen, Kara P.: *Eyetracking Web Usability*. New Riders, 2010

[JS91] Johnson, B. ; Shneidermann, B.: Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In: *VIS '91: Proceedings of the 2nd conference on Visualization '91*, 1991, S. 284–291

[K.04] K., Schwaber: *Agile Project Management with Scrum*. Microsoft Press, 2004

[Kei01] Keim, D. A.: *Visual exploration of large data sets*. Commun, 2001

[KFZ09] Keim, D. A. ; F., Mansmann ; Ziegler, H.: Visual analytics. In: *Encyclopedia of Database Systems* (2009)

[KJ10] Karsten Januszewski, Jaime R.: *The New Iteration*. http://windowsclient.net/wpf/white-papers/thenewiteration.aspx. Version: 2010

[KMS+08] Keim, D. ; Mansmann, F. ; Schneidewind, J. ; Thomas, J. ; Ziegler, H.: *Visual Analytics: Scope and Challenges. In: Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*. Springer, 2008

[Kus09] Kuschel, Mark A.: *Bewertung von AJAX Steuerelementen anhand von Usability-Tests*. GRIN Verlag, 2009

[KW07] Keim, D. ; Ward, M.: *Visualization. In: Berthold, M./Hand, D.: Intelligent Data Analysis: An Introduction*. Springer, 2007

[les09] *WPF Manipulation Basics*. http://blogs.msdn.com/b/llobo/archive/2009/12/21/wpf-manipulation-basics.aspx. Version: 2009. – [Online; Stand 30. März 2010]

[LR96] Lamping, J. ; Rao, R.: The Hyperbolic Browser: A Focus+ Context Technique for Visualizing Large Hierarchies. In: *Journal of Visual Languages and Computing* (1996), S. 33–55

[Mag11] Magdeburg, Uni: *Datenvisualisierung und Data Mininghttp://www.igd.fhg.de/highlights/index.html*. http://fusion.cs.uni-magdeburg.de/pubs/spektrum.pdf. Version: 03 2011

[Mar05] Marchesi, Michele: *The new XP*. http://www.piapetersen.dk/rhs/TheNewXp.pdf. Version: 2005. – Abrufdatum: 4. Juni 2010, originally published at www.agilexp.org

[Maz09] Mazza, R.: *Introduction to information visualization*. 2009

[Mic11a] Microsoft: *Download details: Microsoft(R) Surface(R) Toolkit for Windows Touch Beta*. http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=801907a7-b2dd-4e63-9ff3-8a2e63932a74. Version: 03 2011

[Mic11b] Microsoft: *FAQ | Resources | Microsoft Silverlight*. http://www.microsoft.com/silverlight/faq/. Version: 03 2011

[Mic11c] Microsoft: *The Surface Platform for Real Connections*. http://www.microsoft.com/surface/softwareplatform.aspx. Version: 03 2011

[Mos11] Moser, Christian: *WPF Tutorial | Introduction to WPF*. http://www.wpftutorial.net/WPFIntroduction.html. Version: 03 2011

[MRAK03] Meister, J. ; Rohde, M. ; Appelrath, H.-J. ; Kamp, V.: Data-warehousing im gesundheitswesen. In: *it - Information Technology* (2003)

[Nac08] Nachreiner, Florian: *Usability Testing eines Corporate Intranet. Theorie und Praxis verfügbarer Evaluationsmethoden.* VDM Verlag Dr. Müller Aktiengesellschaft & Co. KG, 2008

[Nie] Nielsen, Jakob: *How to Conduct a Heuristic Evaluation*. http://www.useit.com/papers/heuristic/heuristic_evaluation.html

[Nie05] Nielsen, Jakob: *Ten Usability Heuristics*. http://www.useit.com/papers/heuristic/heuristic_list.html. Version: 2005

[Noc07] Nocke, T.: *Visuelles Data Mining und Visualisierungsdesign für die Klimaforschung*, Universität Rostock, Diss., 2007

[Nor05] North, C.: *Information Visualization*. 2005

[NZ07] Nihad Zehic, Ludwig-Maximilians-Universitaet M. Institut fuer Informatik I. Institut fuer Informatik: *Touchscreen Technologien*. http://www.medien.ifi.lmu.de/lehre/ws0607/mmi1/essays/Nihad-Zehic.xhtml. Version: 04 2007

[OB08] Oestereich B., Weiss S.: *APM Agiles Projektmanagement*. first. 2008

[Old10] Oldenburg, Uni: *Lange Nacht der Wissenschaft an der Universität Oldenburg*. http://www.nacht-der-wissenschaft.uni-oldenburg.de/index.html. Version: 2010

[Old11] Oldenburg, Universität: *Schülerinformationstag Informatik*. http://www.informatik.uni-oldenburg.de/infotag/. Version: 2011

[Rob07] Roberts, C.J.: State of the Art: Coordinated u. Multiple Views in Exploratory Visualization. In: *Proceedings CMV 2007, Washington*, 2007, S. 61–71

[Rog09] Rogowski, Collin: *Agile Software Entwicklung*. 2009. – Veranstaltungsskript SS 2009, Duale Hochschule Baden-Württemberg, Karlsruhe

[S.09] S., Hager: *Vortrag vom klassischem zum agilen Projektmanagement*. http://www.getzcope.com/blog/2009/04/07/vom-klassischen-zum-agilenprojektmanagement-video/. Version: 2009

[Sam11] Samsung: *Samsung Galaxy S I 90000 - Spezifikation - Samsung Mobile Deutschland*. http://www.samsungmobile.de/samsung-handy/samsung-galaxy-s-i9000-specification. Version: 03 2011

[SBH+09] Sedlmair, M. ; Bernhold, C. ; Herrscher, D. ; Boring, S. ; Butz, A.: MostVis: An Interactive Visualization Supporting Automotive Engineers in MOST Catalog Exploration. In: *Proceedings of the 13th International Conference on Information Visualization*, 2009

[Sch07a] Schneidewind, J.: *Scalable Visual Analytics Solutions and Techniques for Business Applications*. KOPS, 2007

[Sch07b] Schumann, H.: *Visual Analytics-quo vadis In: Donau-Universität Krems: Sehen und Verstehen: Bekanntes belegen - Unbekanntes entdecken*. Timnews, 2007

[Sch10a] Schneider, Wolfgang: *Grundsatz Aufgabenangemessenheit*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/aufgabenangemessenheit.htm. Version: 2010

[Sch10b] Schneider, Wolfgang: *Grundsatz Erwartungskonformität*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/erwartungskonformitt.htm. Version: 2010

[Sch10c] Schneider, Wolfgang: *Grundsatz Fehlertoleranz*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/fehlertoleranz.htm. Version: 2010

[Sch10d] Schneider, Wolfgang: *Grundsatz Individualisierbarkeit*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/individualisierbarkeit.htm. Version: 2010

[Sch10e] Schneider, Wolfgang: *Grundsatz Lernförderlichkeit*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/lernfrderlichkeit.htm. Version: 2010

[Sch10f] Schneider, Wolfgang: *Grundsatz Selbstbeschreibungsfähigkeit*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/selbstbeschreibungsfhigkeit_.htm. Version: 2010

[Sch10g] Schneider, Wolfgang: *Grundsatz Steuerbarkeit*. http://www.ergo-online.de/site.aspx?url=html/software/ergonomische_dialoggestaltung/steuerbarkeit.htm. Version: 2010

[Sed08a] Sedlmair, M.: Information visualization for in-car communication processes. In: *Doctoral Colloquium of the 12th International Conference on Information Visualization, London, UK*, 2008

[Sed08b] Sedlmair, M.: MSCar: Enhancing Message Sequence Charts with Interactivity for Analysing (Automotive) Communication Sequences. In: *Proceedings of the 2nd International Workshop on the Layout of (Software) Engineering Diagrams*, 2008

[Sed09] Sedlmair, M.: Using Visual Analytics and Information Visualization to Investigate In-Car Communication Processes. In: *Doctoral Colloquium at IEEE InfoVis/Vast/Vis, VisWeek 2009, Atlantic City, USA, Okt 10, 2009.*, 2009

[Shi08] Shifflett, Karl: *Learning WPF M-V-VM*. Blog entry from 2008/11/08. http://karlshifflett.wordpress.com/2008/11/08/learning-wpf-m-v-vm/. Version: 2008. – Abrufdatum: 22. März 2011

[SHS+08] Sedlmair, M. ; Hintermaier, W. ; Stocker, K. ; Büring, T. ; Butz, A.: A Dual-View Visualization of In-Car Communication Processes. In: *Proceedings of the 12th International Conference on Information Visualization, London, UK*, 2008

[SIBB10] Sedlmair, M. ; Isenberg, P. ; Baur, D. ; Butz, A.: Evaluating Information Visualization in Large Companies: Challenges, Experiences and Recommendations. (2010)

[SKHB09] Sedlmair, M. ; Kunze, B. ; Hintermaier, W. ; Butz, A.: User-centered Development of a Visual Exploration System for In-Car Communication. In: *9th International Symposium on Smart Graphics, SG 2009, Salamanca, Spain, May 28-30, 2009. Proceedings*, 2009

[Smi09] Smith, Josh: WPF Apps With The Model-View-ViewModel Design Pattern. In: *MSDN Magazine* (2009). http://msdn.microsoft.com/en-us/magazine/dd419663.aspx. – Abrufdatum: 22. März 2011

[SRH+09] Sedlmair, M. ; Ruhland, K. ; Hennecke, F. ; Butz, A. ; Bioletti, S. ; O'Sullivan, C.: Towards the Big Picture: Enriching 3D Models with Information Visualisation and Vice Versa. In: *9th International*

Symposium on Smart Graphics, SG 2009, Salamanca, Spain, May 28-30, 2009. Proceedings, 2009

[ST05] Schick, S. ; Theobald, B.: *Informationsvisualisierung im Wissensmanagement*. unknown, 2005

[Sta11] Standardization, ISO International O.: *ISO 9241-210:2010*. http://de.wikipedia.org/w/index.php?title=ISO_13407&oldid=72842252. Version: 2011. – [Online; Stand 25.03.2011]

[TD09] TU-Darmstadt: *Visual Analytics - informelle Kurzeinführung und Beispielanwendungen*. http://www.gris.informatik.tudarmstadt.de/tschreck/vaseminar09/KurzeinfC3BChrung.pdf. Version: 2009. – [Online; Stand 3. Juni 2010]

[tea10] teaching.org e: *Eye Tracking*. http://www.e-teaching.org/didaktik/qualitaet/eye/. Version: 2010

[The10] Thesmann, Stephan: *Einführung in das Design multimedialer Webanwendungen*. Vieweg+Teubner | GWV Fachverlage GmbH, 2010

[TIC09] Tobias, M. ; Isenberg, P. ; Carpendale, S.: *Coordinating co-located collaboration with information visualization*. IEEE Transaction on Visualization and Computer Graphics, 2009

[Tig11] Tigris.org: *tortoisesvn.tigris.org*. http://tortoisesvn.tigris.org/. Version: 03 2011

[Tuf83] Tufte, E: *The visual display of quantitative information*. 1983

[UNK10] UNKNOWN: *Microsoft Visualization Language - The Vedea Project*. http://research.microsoft.com/en-us/projects/vedea/. Version: 2010

[Vis11a] VisMaster: *VisMaster | Visual Analytics - Mastering the Information Age*. http://www.vismaster.eu/faq/the-visual-analytics-process. Version: 03 2011

[Vis11b] VisMaster: *Visual Analytics Process*. http://www.vismaster.eu/faq/the-visual-analytics-process. Version: 03 2011

[Wag04] Wagner, Michael: *Business Networking im Internet. Interaktive Anbahnung von Kooperation in Unternehmensnetzwerken*. Deutscher Universtitäts-Verlag/GWV Fachverlage GmbH, 2004

[Wik10a] Wikipedia: *Flimmerverschmelzungsfrequenz — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Flimmerverschmelzungsfrequenz&oldid=74725014. Version: 2010. – [Online; Stand 09.06.2010]

[Wik10b] Wikipedia: *Gestaltpsychologie — Wikipedia, Die freie Enzyklopädie*. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=52075. Version: 2010. – [Online; Stand 08.06.2010]

[WKD+08] Wassink, I. ; Kulyk, O. ; Dijk, E. van ; Veer, GC van d. ; Vet, PE van d. ; Zudilova-Seinstra, E. ; Adriaansen, T. ; Liere, R. van: Applying a User-centred Approach to Interactive Visualization Design. In: *Trends in Interactive Visualization* (2008), S. 175–199

[WMW09a] Wobbrock, Jacob O. ; Morris, Meredith R. ; Wilson, Andrew D.: User-Defined Gestures for Surface Computing. In: *Proceedings of the 27th international conference on Human factors in computing systems*. New York, NY, USA, 2009

[WMW09b] Wobbrock, J.O. ; Morris, M.R. ; Wilson, A.D.: User-defined gestures for surface computing. In: *Proceedings of the 27th international conference on Human factors in computing systems* ACM, 2009, S. 1083–1092

[XLS10] XLSTAT: *Wie erhalte ich eine Visualisierung in parallelen Koordinaten?* http://www.xlstat.com/de/support/tutorials/pcor.htm. Version: 2010. – [Online; Stand 4. Juni 2010]

[Züh04] Zühlke, Detlef: *Useware-Engineering für technische Systeme*. Springer Verlag, 2004

[Zic09] Zick, Jürgen: *IT-System- und Netzwerkmanagement. Vom Groß- zum Kleinbetrieb. Zwei Usability-Studien.* kassel university press GmbH, 2009